# A Complete Subsumption Algorithm

Stefano Ferilli, Nicola Di Mauro, Teresa M.A. Basile, and Floriana Esposito

Dipartimento di Informatica, Università di Bari
via E. Orabona, 4, 70125 Bari, Italia
{ferilli,nicodimauro,basile,esposito}@di.uniba.it

**Abstract.** Efficiency of the first-order logic proof procedure is a major issue when deduction systems are to be used in real environments, both on their own and as a component of larger systems (e.g., learning systems). Hence, the need of techniques that can perform such a process with reduced time/space requirements (specifically when performing resolution). This paper proposes a new algorithm that is able to return the whole set of solutions to $\theta$-subsumption problems by compactly representing substitutions. It could be exploited when techniques available in the literature are not suitable. Experimental results on its performance are encouraging.

## 1 Introduction

The classical Logic Programming [8] provability relation, logic implication, has been shown to be undecidable [12], which is too strict a bias to be accepted. Hence, a weaker but decidable generality relation, called $\theta$-subsumption, is often used in practice. Given $C$ and $D$ clauses, $C$ $\theta$-subsumes $D$ (often written $C \leq D$) iff there is a substitution $\theta$ such that $C\theta \subseteq D$. A substitution is a mapping from variables to terms, often denoted by $\theta = \{X_1 \rightarrow t_1, \ldots, X_n \rightarrow t_n\}$, whose application to a clause $C$, denoted by $C\theta$, rewrites all the occurrences of variables $X_i$ $(i = 1 \ldots n)$ in $C$ by the corresponding term $t_i$. Thus, since program execution corresponds to proving a theorem, efficiency of the generality relation used is a key issue that deserves great attention.

In the following, we will assume that $C$ and $D$ are Horn clauses having the same predicate in their head, and that the aim is checking whether $C$ $\theta$-subsumes $D$. Note that $D$ can always be considered ground (i.e., variable-free) without loss of generality. Indeed, in case it is not, each of its variables can be replaced by a new constant not appearing in $C$ nor in $D$ (*skolemization*), and it can be proven that $C$ $\theta$-subsumes $D$ iff $C$ $\theta$-subsumes the skolemization of $D$. $|\cdot|$ will denote, as usual, the cardinality of a set (in particular, when applied to a clause, it will refer to the number of literals composing it). Since testing if $C$ $\theta$-subsumes $D$ can be cast as a refutation of $\{C\} \cup \neg D$, a basic algorithm can be obtained in Prolog by skolemizing $D$, then asserting all the literals in the body of $D$ an the clause $C$, and finally querying the head of $D$. The outcome is computed by Prolog through SLD resolution [10], which can be very inefficient under some conditions, as for $C$ and $D$ in the following example.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 1–13, 2003.
© Springer-Verlag Berlin Heidelberg 2003

*Example 1.* $C$ = `h(`$X_1$`) :- p(`$X_1$`,`$X_2$`),p(`$X_2$`,`$X_3$`),...,p(`$X_{n-1}$`,`$X_n$`),q(`$X_n$`).`
$D$ = `h(`$c_1$`) :- p(`$c_1$`,`$c_1$`), p(`$c_1$`,`$c_2$`), ..., p(`$c_1$`,`$c_n$`), p(`$c_2$`,`$c_1$`),`
`p(`$c_2$`,`$c_2$`), ..., p(`$c_2$`,`$c_n$`), ..., p(`$c_n$`,`$c_1$`), p(`$c_n$`,`$c_2$`), ..., p(`$c_n$`,`$c_n$`).`

In the following, the next section presents past work in this field; Section 3 presents the new $\theta$-subsumption algorithm, and Section 4 shows experimental results concerning its performance. Lastly, Section 5 concludes the paper.

## 2 Related Work

The great importance of finding efficient $\theta$-subsumption algorithms is reflected by the amount of work carried out so far in this direction in the literature. In the following, we briefly recall some milestones in this research field.

Our brief survey starts from Gottlob and Leitsch [5]. After investigating two classical algorithms (by Chang & Lee [1] and Stillman [14]) in order to assess their worst-case time complexity, based on the number of performed literal unifications, they define a new backtracking algorithm that attacks the problem complexity through a *divide and conquer* strategy, by first partitioning the clause into independent subsets, and then applying resolution separately to each of them, additionally exploiting a heuristic that resolves each time the literal with the highest number of variables that occur also in other literals.

A more formal approach was then taken by Kietz and Lübbe in [7]. They start from the following definition:

**Definition 1.** *Let $C = C_0 \leftarrow C_{Body}$ and $D = D_0 \leftarrow D_{Body}$ be Horn clauses. $C$* deterministically *$\theta$-subsumes $D$, written $C \vdash_{\theta DET} D$, by $\theta = \theta_0\theta_1 \ldots \theta_n$ iff $C_0\theta_0 = D_0$ and there exists an ordering $C'_{Body} = C_1, \ldots, C_n$ of $C_{Body}$ such that for all $i$, $1 \leq i \leq n$, there exists exactly one $\theta_i$ such that $\{C_1, \ldots, C_i\}\theta_0\theta_1 \ldots \theta_i \subseteq D_{Body}$.*

Since in general $C \not\vdash_{\theta DET} D$, in addition to identifying the subset of $C$ that deterministically $\theta$-subsumes $D$, $C_{DET}$, the algorithm can also return the rest of $C$, $C_{NONDET}$, to which other techniques can be applied according to the definition of *non-determinate locals*, corresponding to the independent parts of $C_{NONDET}$ according to Gottlob and Leitsch. They can be identified in polynomial time, and handled separately by $\theta$-subsumption algorithms.

The above ideas were extended by Scheffer, Herbrich and Wysotzki [11] by transposing the problem into a graph framework, in which additional techniques can be exploited. First, the authors extend the notion of 'determinism' in matching candidates by taking into account not just single literals, but also their 'context' (i.e., the literals to which they are connected via common variables). Indeed, by requiring that two literals have the same context in order to be matched, the number of literals in $C$ that have a unique matching candidate in $D$ potentially grows. Taking into account the context allows to test for subsumption in polynomial time a proper superset of the set of determinate clauses according to the definition by Kiets and Lübbe. The remaining (non-determinate) part of $C$ is

then handled by mapping the subsumption problem onto a search for the maximum clique in a graph, for which known efficient algorithms can be exploited, properly tailored.

In sum, all the work described so far can be condensed in the following algorithm: First the 'extended' (according to the context definition) determinate part of $C$ and $D$ is matched; then the locals are identified, and each is attacked separately by means of the clique algorithm. Note that all the proposed techniques rely on backtracking, and try to limit its effect by properly choosing the candidates in each tentative step. Hence, all of them return only the first subsuming substitution found, even if many exist.

Finally, Maloberti and Sebag in [9] face the problem of $\theta$-subsumption by mapping it onto a Constraint Satisfaction Problem (CSP). Different versions of a correct and complete $\theta$-subsumption algorithm, named *Django*, were built, each implementing different (combinations of) CSP heuristics. Experiments are reported, proving a difference in performance of several orders of magnitude in favor of Django compared to the algorithms described above. Note that Django only gives a binary (*yes* or *no*) answer to the subsumption test, without providing any matching substitution in case of positive outcome.

## 3 A New Approach

Previous successful results obtained on the efficiency improvement of the matching procedure under the Object Identity framework [3] led us to extend those ideas to the general case. The main idea to avoid backtracking and build in one step the whole set of subsumption solutions is to compress information on many substitutions by compactly representing them in a single structure. For this reason, some preliminary definitions are necessary.

### 3.1 Preliminaries

Let us start by recalling a useful definition from the literature.

**Definition 2 (Matching Substitution).** *A* matching Substitution *from a literal $l_1$ to a literal $l_2$ is a substitution $\mu$, such that $l_1\mu = l_2$.*

The set of all matching substitutions from a literal $l \in C$ to some literal in $D$ is denoted by [2] $uni(C, l, D) = \{\mu \mid l \in C, l\mu \in D\}$

Now, it is possible to define the structure to compactly represent sets of substitutions.

**Definition 3 (Multisubstitutions).** *A* multibind *is denoted by $X \rightarrow T$, where $X$ is a variable and $T \neq \emptyset$ is a set of constants. A* multisubstitution *is a set of multibinds $\Theta = \{X_1 \rightarrow T_1, \ldots, X_n \rightarrow T_n\} \neq \emptyset$, where $\forall i \neq j : \ X_i \neq X_j$.*

In particular, a single substitution is represented by a multisubstitution in which each constants set is a singleton ($\forall i : \ \mid T_i \mid \ = 1$). In the following, multisubstitutions will be denoted by capital Greek letters, and normal substitutions by lower-case Greek letters.

*Example 2.* $\Theta = \{X \to \{1,3,4\}, Y \to \{7\}, Z \to \{2,9\}\}$ is a multisubstitution. It contains 3 multibinds, namely: $X \to \{1,3,4\}$, $Y \to \{7\}$ and $Z \to \{2,9\}$.

**Definition 4 (Split).** *Given a multisubstitution* $\Theta = \{X_1 \to T_1, \ldots, X_n \to T_n\}$, split($\Theta$) *is the set of all substitutions represented by* $\Theta$*:*
*split*$(\Theta) = \{\ \{X_1 \to c_{i_1}, \ldots, X_n \to c_{i_n}\}\ | \forall k = 1 \ldots n : c_{i_k} \in T_k \wedge i = 1 \ldots |T_k|\}$.

*Example 3.* Let us find the set of all substitutions represented by the multisubstitution $\Theta = \{X \to \{1,3,4\}, Y \to \{7\}, Z \to \{2,9\}\}$, *split*$(\Theta) = \{\{X \to 1, Y \to 7, Z \to 2\}, \{X \to 1, Y \to 7, Z \to 9\}, \{X \to 3, Y \to 7, Z \to 2\}, \{X \to 3, Y \to 7, Z \to 9\}, \{X \to 4, Y \to 7, Z \to 2\}, \{X \to 4, Y \to 7, Z \to 9\}\}$

**Definition 5 (Union of Multisubstitutions).** *The union of two multisubstitutions* $\Theta' = \{\overline{X} \to T', X_1 \to T_1, \ldots, X_n \to T_n\}$ *and* $\Theta'' = \{\overline{X} \to T'', X_1 \to T_1, \ldots, X_n \to T_n\}$ *is the multisubstitution*

$$\Theta' \sqcup \Theta'' = \{\overline{X} \to T' \cup T''\} \cup \{X_i \to T_i\}_{1 \le i \le n}$$

Note that the two input multisubstitutions must be defined on the same set of variables and must differ in at most one multibind.

*Example 4.* The union of two multisubstitutions $\Sigma = \{X \to \{1,3\}, Y \to \{7\}, Z \to \{2,9\}\}$ and $\Theta = \{X \to \{1,4\}, Y \to \{7\}, Z \to \{2,9\}\}$, is:

$$\Sigma \sqcup \Theta = \{X \to \{1,3,4\}, Y \to \{7\}, Z \to \{2,9\}\}$$

(the only different multibinds being those referring to variable $X$).

**Definition 6 (Merge).** *Given a set* $\mathcal{S}$ *of substitutions on the same variables,* merge($\mathcal{S}$) *is the set of multisubstitutions obtained according to Algorithm 1.*

*Example 5.* merge($\{\{X \to 1, Y \to 2, Z \to 3\}, \{X \to 1, Y \to 2, Z \to 4\}, (X \to 1, Y \to 2, Z \to 5\}\}) =$ merge($\{\{X \to \{1\}, Y \to \{2\}, Z \to \{3,4\}\}, \{X \to \{1\}, Y \to \{2\}, Z \to \{5\}\}\}) = \{\{X \to \{1\}, Y \to \{2\}, Z \to \{3,4,5\}\}\}$. This way we can represent 3 substitutions with only one multisubstitution.

**Definition 7 (Intersection of Multisubstitutions).** *The intersection of two multisubstitutions* $\Sigma = \{X_1 \to S_1, \ldots, X_n \to S_n, Y_1 \to S_{n+1}, \ldots, Y_m \to S_{n+m}\}$ *and* $\Theta = \{X_1 \to T_1, \ldots, X_n \to T_n, Z_1 \to T_{n+1}, \ldots, Z_l \to T_{n+l}\}$*, where* $n, m, l \ge 0$ *and* $\forall j, k : Y_j \ne Z_k$*, is the multisubstitution defined as:*
$\Sigma \sqcap \Theta = \{X_i \to S_i \cap T_i\}_{i=1\ldots n} \cup \{Y_j \to S_{n+j}\}_{j=1\ldots m} \cup \{Z_k \to T_{n+k}\}_{k=1\ldots l}$
*iff* $\forall i = 1 \ldots n : S_i \cap T_i \ne \emptyset$*; otherwise it is undefined.*

**Algorithm 1 merge($\mathcal{S}$)**

**Require:** $\mathcal{S}$: set of substitutions (each represented as a multisubstitution)
  **while** $\exists u, v \in \mathcal{S}$ such that $u \ne v$ and $u \sqcup v = t$ **do**
    $\mathcal{S} := (\mathcal{S} \setminus \{u, v\}) \cup \{t\}$
  **end while**
  return $\mathcal{S}$

*Example 6.* The intersection of two multisubstitutions $\Sigma = \{X \to \{1,3,4\}, Z \to \{2,8,9\}\}$ and $\Theta = \{Y \to \{7\}, Z \to \{1,2,9\}\}$ is: $\Sigma \sqcap \Theta = \{X \to \{1,3,4\}, Y \to \{7\}, Z \to \{2,9\}\}$. The intersection of $\Sigma = \{X \to \{1,3,4\}, Z \to \{8,9\}\}$ and $\Theta = \{Y \to \{7\}, Z \to \{1,2\}\}$ is undefined.

**Lemma 1.** *The $\sqcap$ operator is monotonic in the set of variables. Specifically,*

$$|\Sigma|, |\Theta| \leq |\Sigma \sqcap \Theta| = n + m + l$$

*Proof.* The $\sqcap$ operator transposes in the result all the multibinds concerning $Y_j, j = 1 \ldots m$ variables from $\Sigma$, and all the multibinds concerning $Z_k, k = 1 \ldots l$ variables from $\Theta$, whose constant sets are all nonempty by definition. Moreover, it preserves all the multibinds concerning $X_i, i = 1 \ldots n$ variables common to $\Sigma$ and $\Theta$, since all intersections of the corresponding constants sets must be nonempty for the result to be defined. Hence: $n, m, l \geq 0$ and $\forall j, k : \ Y_j \neq Z_k$ implies that $|\Sigma \sqcap \Theta| = n + m + l$ and both $|\Sigma| = n + m \leq |\Sigma \sqcap \Theta|$ and $|\Theta| = n + l \leq |\Sigma \sqcap \Theta|$.

The above $\sqcap$ operator is able to check if two multisubstitutions are compatible (i.e., if they share at least one of the substitutions they represent). Indeed, given two multisubstitutions $\Sigma$ and $\Theta$, if $\Sigma \sqcap \Theta$ is undefined, then there must be at least one variable $X$, common to $\Sigma$ and $\Theta$, to which the corresponding multibinds associate disjoint sets of constants, which means that it does not exist a constant to be associated to $X$ by both $\Sigma$ and $\Theta$, and hence a common substitution cannot exist as well.

The $\sqcap$ operator can be extended to the case of sets of multisubstitutions. Specifically, given two sets of multisubstitutions $\mathcal{S}$ and $\mathcal{T}$, their intersection is defined as the set of multisubstitutions obtained as follows:

$$\mathcal{S} \sqcap \mathcal{T} = \{\Sigma \sqcap \Theta \mid \Sigma \in \mathcal{S}, \Theta \in \mathcal{T}\}$$

Note that, whereas a multisubstitution (and hence an intersection of multisubstitutions) is or is not defined, but cannot be empty, a set of multisubstitutions can be empty. Hence, an intersection of sets of multisubstitutions, in particular, can be empty (which happens when all of its composing intersections are undefined).

### 3.2 The Matching Algorithm

In the following, for the sake of readability, we use the expression $\theta \in \mathcal{T}$ to say that the substitution $\theta$ belongs to the split of some multisubstitution in the set of multisubstitutions $\mathcal{T}$.

**Proposition 1.** $\forall \theta : C\theta \subseteq D \Leftrightarrow \theta \in \mathcal{S}_n$.

*Proof.* Let $C = \{l_1, \ldots, l_n\}$ and $\forall i = 1 \ldots n : \mathcal{T}_i = merge(uni(C, l_i, D))$; let $\mathcal{S}_1 = \mathcal{T}_1$ and $\forall i = 2 \ldots n : \mathcal{S}_i = \mathcal{S}_{i-1} \sqcap \mathcal{T}_i$.
($\Leftarrow$) By induction on $i$: $\forall i \in \{1, \ldots, n\} : \mathcal{S}_i \neq \emptyset \Rightarrow \forall \theta \in \mathcal{S}_i : \{l_1, \ldots, l_i\}\theta \subseteq D$.
**Base** $\emptyset \neq \mathcal{S}_1 = \mathcal{T}_1 \Rightarrow \forall \theta \in \mathcal{T}_1 : \exists k \in D \ni l_1\theta = k \in D \Rightarrow \{l_1\}\theta = \{k\} \subseteq D$.
**Step** $\mathcal{S}_i = \mathcal{S}_{i-1} \sqcap \mathcal{T}_i \neq \emptyset \Rightarrow$ (by definition of $\sqcap$) $\exists \Sigma \in \mathcal{S}_{i-1}, \Theta \in \mathcal{T}_i \ni' \Sigma \sqcap \Theta$

**Algorithm 2 matching**$(C, D)$

**Require:** $C : c_0 \leftarrow c_1, c_2, \ldots, c_n$, $D : d_0 \leftarrow d_1, d_2, \ldots, d_m$: clauses
  **if** $\exists \theta_0$ substitution such that $c_0\theta_0 = d_0$ **then**
    $S_0 := \{\theta_0\}$;
    **for** $i := 1$ to $n$ **do**
      $S_i := S_{i-1} \sqcap merge(uni(C, c_i, D))$
    **end for**
  **end if**
  return $(S_n \neq \emptyset)$

defined $\Rightarrow \forall \gamma \in \Sigma \sqcap \Theta : \gamma = \sigma\theta \nexists' \sigma \in split(\Sigma), \theta \in split(\Theta) \wedge \sigma, \theta$ compatible $\Rightarrow \{l_1, \ldots, l_{i-1}\}\sigma \subseteq D$ (by hypothesis) $\wedge \{l_i\}\theta \subseteq D$ (by definition of $\mathcal{T}_i$) $\Rightarrow \{l_1, \ldots, l_{i-1}\}\sigma \cup \{l_i\}\theta \subseteq D \Rightarrow \{l_1, \ldots, l_i\}\sigma\theta \subseteq D$. This holds, in particular, for $i = n$, which yields the thesis.
($\Rightarrow$) By induction on $i$: $\forall i \in \{1, \ldots, n\} : \{l_1, \ldots, l_i\}\theta \subseteq D \Rightarrow \theta \in \mathcal{S}_i$.
**Base** (*Ad absurdum*) $\nexists \theta|_{\{l_1\}} \Rightarrow \theta|_{\{l_1\}} \notin merge(uni(C, l_1, D)) \Rightarrow \theta|_{\{l_1\}} \notin uni(C, l_1, D) \Rightarrow \{l_1\}\theta|_{\{l_1\}} \notin D \Rightarrow \{l_1\}\theta \notin D$. But $\{l_1\}\theta \in D$ by hypothesis.
**Step** (*Ad absurdum*) $\theta|_{\{l_1,\ldots,i\}} (= \theta|_{\{l_1,\ldots,l_{i-1}\}}\theta|_{\{l_i\}}) \notin \mathcal{S}_i$. By construction, $\mathcal{S}_i = \mathcal{S}_{i-1} \sqcap \mathcal{T}_i$. By inductive hypothesis, $\theta|_{\{l_1,\ldots,l_{i-1}\}} \in \mathcal{S}_{i-1}$. Thus, $\theta|_{\{l_i\}} \notin \mathcal{T}_i \Rightarrow \{l_i\}\theta|_{\{l_i\}} \notin D \Rightarrow \{l_i\}\theta \notin D$. But, by hypothesis, $\{l_1, \ldots, l_i\}\theta \subseteq D \Rightarrow \{l_i\}\theta \in D$.

This leads to the $\theta$-subsumption procedure reported in Algorithm 2. It should be noted that the set of multisubstitutions resulting from the merging phase could be not unique. In fact, it may depend on the order in which the two multisubstitutions to be merged are chosen at each step. The presented algorithm does not currently specify any particular principle according to which performing such a choice, but this issue is undoubtedly a very interesting one, and deserves a specific study (that is outside the scope of this paper) in order to understand if the compression quality of the result is actually affected by the ordering and, in such a case, if there are heuristics that can suggest in what order the multisubstitutions to be merged have to be taken in order to get an optimal result.

*Example 7.* Consider the following substitutions:
$\theta = \{X \leftarrow 1, Y \leftarrow 2, Z \leftarrow 3\}$ $\quad\quad$ $\delta = \{X \leftarrow 1, Y \leftarrow 2, Z \leftarrow 4\}$
$\sigma = \{X \leftarrow 1, Y \leftarrow 2, Z \leftarrow 5\}$ $\quad\quad$ $\tau = \{X \leftarrow 1, Y \leftarrow 5, Z \leftarrow 3\}$
One possible merging sequence is $(\theta \sqcup \delta) \sqcup \sigma$, that prevents further merging $\tau$ and yields the following set of multisubstitutions:

$$\{\{X \leftarrow \{1\}, Y \leftarrow \{2\}, Z \leftarrow \{3, 4, 5\}\}, \{X \leftarrow \{1\}, Y \leftarrow \{5\}, Z \leftarrow \{3\}\}\}$$

Another possibility is first merging $\theta \sqcup \tau$ and then $\delta \sqcup \sigma$, that cannot be further merged and hence yield:

$$\{\{X \leftarrow \{1\}, Y \leftarrow \{2, 5\}, Z \leftarrow \{3\}\}, \{X \leftarrow \{1\}, Y \leftarrow \{2\}, Z \leftarrow \{4, 5\}\}\}$$

### 3.3 Discussion

Ideas presented in related work aimed, in part, at leveraging on particular situations in which the $\theta$-subsumption test can be computed with reduced complexity. This aim inspired, for instance, the concepts of *determinate* (part of a) clause and of *k-locals*. However, after identifying such determinate and independent subparts of the given clauses, the only possible way out is applying classical, complex algorithms, possibly exploiting heuristics to choose the next literal to be unified. In those cases, the CSP approach proves very efficient, but at the cost of not returning (all the) possible substitutions by which the matching holds. Actually, there are cases in which at least one such substitution is needed by the experimenter. Moreover, if *all* such substitutions are needed (e.g., for performing successive resolution steps), the feeling is that the CSP approach has to necessarily explore the whole search space, thus loosing all the advantages on which it bases its efficiency. The proposed algorithm, on the contrary, returns *all* possible matching substitutions, without performing any backtracking in their computation. Specifically, its search strategy consists in a kind of breadth-first in which the explored nodes of the search space are compressed; this means that, when no compression is possible for the substitutions of each literal, it becomes a normal breadth-first search (it would be interesting to discuss in what – non purposely designed – situations it happens). Hence, it is worth discussing the complexity of the different steps involved thereof. Because of the above considerations, in the following only linked clauses will be taken into account, so that neither determinate matching nor partitioning into $k$-locals apply.

Let $p_i$ be the $i$-th distinct predicate in $C$, $a_i$ its arity and $m_i$ be the number of literals in $D$ with predicate symbol $p_i$. Let $l_j$ be the $j$-th literal in $C$. Call $a$ the maximum arity of predicates in $C$ (predicates with greater arity in $D$ would not be considered for matching), and $c$ the number of distinct constants in $D$. Each unifier of a given $p_i$ with a literal on the same predicate symbol in $D$ can be computed in $a_i$ steps. There are $m_i$ such unifiers to be computed (each represented as a multisubstitution), hence computing $uni(C, l, D)$ has complexity $a_i * m_i$ for any literal $l \in C$ built on predicate $p_i$. Note that the constants associated to each argument of $p_i$ are the same for all literals in $C$ built on it, hence such a computation can be made just once for each different predicate, and then tailored to each literal by just changing the variables in each multibind.

(Checking and) merging two multisubstitutions requires them to differ in at most one multibind (as soon as two different multibinds are found in the two multisubstitutions, the computation of their merging stops with failure). Hence, the complexity of merging two multisubstitutions is less than $a_i * 2m_i$, since there are at most $a_i$ arguments to be checked, each made up of at most $m_i$ constants (one for each compatible literal, in case they are all different)[1]. The multisubstitutions in the set $uni(C, l, D)$ can be merged by pairwise comparing (and, possibly, merging) any two of them, and further repeating this on the new sets stepwise obtained, until no merging is performed or all multisubstitutions have been merged

[1] Assuming that the constants in each multibind are sorted, checking the equality of two multibinds requires to scan each just once.

into one. At the $k$-th step ($0 \leq k \leq m_i - 1$), since at least one merging was performed at each step, the set will contain at most $m_i - k$ multisubstitutions, for a total of at most $\binom{m_i-k}{2}$ couples to be checked and (possibly) merged. Globally, we have a merge complexity equal to[2] $\sum_{k=1}^{m_i-1} \binom{m_i-k}{2} * a_i * 2m_i \sim O(a_i * m_i^4)$.

As to the intersection between two multisubstitutions, note that one of the two refers to a literal $l \in C$ built on a predicate $p_i$, and hence will be made up of $a_i$ multibinds, each of at most $m_i$ constants. In the (pessimistic) case that all of the variables in $l$ are present in the other multisubstitution, the complexity of the intersection is therefore[3] $a_i * m_i * min(c, |D|)$.

When discussing the overall complexity of the whole procedure, it is necessary to take into account that a number of interrelations exist among the involved objects, such that a growth of one parameter often corresponds to the decrease of another. Thus, this is not a straightforward issue. Nevertheless, one intuitive worst case is when non merging can take place among substitutions for all literals[4] and each substitution of any literal is compatible with any substitution of all the others. In such a case, the number of intersections is $O(m^n)$ (supposing each literal in $C$ has $m$ matching substitutions in $D$), but it should be noted that in this case each intersection does not require any computation and is reduced to just an append operation. One intuitive best case is when all substitutions for each literal can be merged into one. In this case, the dominant complexity is that of merging, i.e. $O(n * a * m^4)$.

## 4 Experiments

The new algorithm has been implemented in C, and its performance in computing $\theta$-subsumption between Horn clauses having the same predicate in their head was to be assessed. Actually, to the authors' knowledge, no algorithm is

[2] Actually, this is a pessimistic upper bound, that will not ever be reached. Indeed, it is straightforward to note that a number of simplifying interrelations (not taken into account here for simplicity) hold: e.g., the number of steps is *at most* $m_i$; the more the number of performed mergings, the less the number of possible steps, and the less the number of substitutions to be merged at each step; the more the number of steps, the less the number of merged multisubstitutions, and the less the number of constants in each of them; at each step, only the new merged multisubstitutions are to be taken into account for merging with the previous ones; and so on.

[3] Note that the other multisubstitution comes from past intersections of multisubsitutions referred to already processed literals, and hence each of its multibinds may contain at most a number of constants equal to the maximum number of literals in $D$ that are compatible with a literal in $C$, i.e. $max_i(m_i) \leq |D|$, or to the maximum number of different constants, whichever is the less: $min(c, |D|)$.

[4] Remember that we suppose to deal only with linked clauses, otherwise the matching procedure can be applied separately to the single connected components and then the global substitution can be obtained by simply combining in all possible ways such partial solutions, that will be obviously compatible since they do not share any variable.

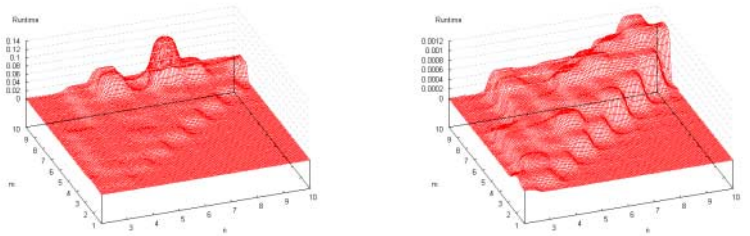



**Fig. 1.** Performance of the new algorithm and Django on Mutagenesis (sec)

available that computes the whole set of substitutions (except forcing all possible backtrackings in the previous ones, which could yield unacceptable runtimes), to which comparing the proposed one. Thus, the choice was between not making a comparison at all, or comparing the new algorithm to Django (the best-performing among those described in Section 2). In the second case, it is clear that the challenge was not completely fair for our algorithm, since it *always* computes the whole set of solutions, whereas Django computes none (it just answers 'yes' or 'no'). Nevertheless, the second option was preferred, according to the principle that a comparison with a faster system could in any case provide useful information on the new algorithm performance, if its handicap is properly taken into account. The need for downward-compatibility in the system output forced to translate the new algorithm's results in the lower-level answers of Django, and hence to interpret them just as 'yes' (independently of how many substitutions were computed, which is very unfair for our algorithm) or 'no' (if no subsuming substitution exists). Hence, in evaluating the experimental results, one should take into account such a difference, so that a slightly worse performance of the proposed algorithm with respect to Django should be considered an acceptable tradeoff for getting all the solutions whenever they are required by the experimental settings. Of course, the targets of the two algorithms are different, and it is clear that in case a binary answer is sufficient the latter should be used.

A first comparison was carried out on a task exploited for evaluating Django by its Authors: The *Mutagenesis* problem [13]. The experiment was run on a PC platform equipped with an Intel Celeron 1.3 GHz processor and running the Linux operating system. In the Mutagenesis dataset, artificial hypotheses were generated according to the procedure reported in [9]. For given $m$ and $n$, such a procedure returns an hypothesis made up of $m$ literals $bond(X_i, X_j)$ and involving $n$ variables, where the variables $X_i$ and $X_j$ in each literal are randomly selected among $n$ variables $\{X_1, \ldots, X_n\}$ in such a way that $X_i \neq X_j$ and the overall hypothesis is linked [6]. The cases in which $n > m + 1$ were not considered, since it is not possible to build a clause with $m$ binary literals that contains more than $m + 1$ variables and that fulfills the imposed linkedness constraint. Specifically, for each $(m, n)$ pair $(1 \leq m \leq 10, 2 \leq n \leq 10)$, 10 artificial hypothe-

**Table 1.** Mean time on the Mutagenesis problem for the three algorithms (sec)

| SLD | Matching | Django |
|---|---|---|
| 158,2358 | 0,01880281 | 0,00049569 |

ses were generated and each was checked against all 229 examples provided in the Mutagenesis dataset. Then, the mean performance of each hypothesis on the 229 examples was computed, and finally the computational cost for each $(m, n)$ pair was obtained as the average $\theta$-subsumption cost over all the times of the corresponding 10 hypotheses.

Figure 1 reports the performance obtained by our algorithm and by Django (respectively) on the $\theta$-subsumption tests for the Mutagenesis dataset. Timings are measured in seconds. The shape of Django's performance plot is smoother, while that of the proposed algorithm shows sharper peaks in a generally flat landscape. The proposed algorithm, after an initial increase, suggests a decrease in computational times for increasing values of $n$ (when $m$ is high). It is noticeable that Django shows an increasingly worse performance on the diagonal[5], while there is no such phenomenon in the plot on the left of Figure 1. However, there is no appreciable difference in computational times, since both systems stay far below the 1 sec threshold.

Table 1 reports the mean time on the Mutagenesis Problem for the three algorithms to get the answer (backtracking was forced in SLD in order to obtain all the solutions). It is possible to note that the Matching algorithm is $8415,5$ times more efficient than the SLD procedure (such a comparison makes no sense for Django because it just answers 'yes' or 'no'). To have an idea of the effort spent, the mean number of substitutions was $91,21$ (obviously, averaged only on positive tests, that are $8,95\%$ of all cases).

Another interesting task concernes *Phase Transition* [4], a particularly hard artificial problem purposely designed to study the complexity of matching First Order Logic formulas in a given universe in order to find their models, if any. A number of pairs clause-example were generated according to the guidelines reported in [4]. Like in [9], $n$ was set to 10, $m$ ranges in $[10, 60]$ (actually, a wider range than in [9]) and $L$ ranges in $[10, 50]$. To limit the total computational cost, $N$ was set to 64 instead of 100: This does not affect the presence of the phase transition phenomenon, but just causes the number of possible substitutions to be less. For each pair $(m, L)$, 33 pairs $(hypothesis, example)$ were constructed, and the average $\theta$-subsumption computational cost was computed as the seconds required by the two algorithms. Both show their peaks in correspondence of low values of $L$ and/or $m$, but such peaks are more concentrated and abruptly rising in the new algorithm. Of course, there is an orders-of-magnitude difference between the two performances (Django's highest peak is 0.037 sec, whereas our

[5] Such a region corresponds to hypotheses with $i$ literals and $i + 1$ variables. Such hypotheses are particularly challenging for the $\theta$-subsumption test since their literals form a chain of variables (because of linkedness).

**Table 2.** Average $\theta$-subsumption cost in the YES, NO an PT regions

| | | NO | Phase Transition | YES | NEG |
|---|---|---|---|---|---|
| Django | Mean | 0,003907 | 0,00663 | 0,005189 | 0,003761 |
| | St-Dev | 0,004867 | 0,00756 | 0,004673 | 0,00455 |
| Matching | Mean | 0,1558803 | 3,5584 | 7,5501 | 0,1139 |
| | St-Dev | 0,75848 | 10,5046 | 20,954 | 0,5147 |
| | | 39,8977 | 536,7119 | 1455,02023 | 30,2845 |

algorithm's top peak is 155.548 sec), but one has to take into account that the new algorithm also returns the whole set of substitutions (if any, which means that a 'yes' outcome may in fact hide a huge computational effort when the solutions are very dense), and it almost does this in reasonable time (only 5.93% of computations took more than 1 sec, and only 1.29% took more than 15 sec).

The mean $\theta$-subsumption costs in various regions are summarized in Table 2. The region is assigned to a problem $(m, L)$ according to the fraction $f$ of clauses $\mathcal{C}$ subsuming examples $Ex$, over all pairs $(\mathcal{C}, Ex)$ generated for that problem. In particular, $f > 90\%$ denotes YES region, $10\% \leq f \leq 90\%$ denotes PT region, and $f < 10\%$ means NO region. While for Django the cost in PT is 1,7 times the cost in NO and 1,3 times the cost in YES, thus confirming the difficulty of that region, in the new algorithm the cost across the regions grows according to the number of substitutions, as expected. The last column reports the cost in a region (NEG) corresponding to the particular case $f = 0\%$ (i.e., there are no substitutions at all). The last row shows the gain of Django over the new algorithm. Again, as expected, the gain grows as the number of solution increases, because it stops immediately after getting an answer, whereas the new algorithm continues until all substitutions are found. The only region in which a comparison is feasible is NEG, where Django is 30 times better than Matching (this could be improved by introducing heuristics that can bias our algorithm towards recognizing a negative answer as soon as possible).

## 5 Conclusions and Future Work

This paper proposed a new algorithm for computing the whole set of solutions to $\theta$-subsumption problems, whose efficiency derives from a proper representation of substitutions that allows to avoid backtracking (which may cause, in particular situations, unacceptable growth of computational times in classical subsumption mechanisms). Experimental results suggest that it is able to carry out its task with high efficiency.

Actually, it is not directly comparable to other state-of-the-art systems, since its characteristic of yielding all the possible substitution by which $\theta$-subsumption holds has no competitors. Nevertheless, a comparison seemed useful to get an idea of the cost in time performance for getting such a plus. The good news is that, even on hard problems, and notwithstanding its harder computational

effort, the new algorithm turned out to be in most cases comparable, and in any case at least acceptable, with respect to the best-performing system in the literature. A Prolog version of the algorithm is currently used in a system for inductive learning from examples.

Future work will concern an analysis of the complexity of the presented algorithm, and the definition of heuristics that can further improve its efficiency (e.g., heuristics that may guide the choice of the best literal to choose at any step in order to recognize as soon as possible the impossibility of subsumption).

## Acknowledgements

This work was partially funded by the EU project IST-1999-20882 COLLATE. The authors would like to thank Michele Sebag and Jerome Maloberti for making available the Django and for the suggestions on its use, and the anonymous reviewers for useful comments.

## References

[1] C. L. Chang and R. C. T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973. 2

[2] N. Eisinger. Subsumption and connection graphs. In J. H. Siekmann, editor, *GWAI-81, German Workshop on Artificial Intelligence, Bad Honnef, January 1981*, pages 188–198. Springer, Berlin, Heidelberg, 1981. 3

[3] S. Ferilli, N. Fanizzi, N. Di Mauro, and T. M. A. Basile. Efficient $\theta$-subsumption under Object Identity. In *Atti del Workshop AI*IA su Apprendimento Automatico, Siena - Italy*, 2002. 3

[4] A. Giordana, M. Botta, and L. Saitta. An experimental study of phase transitions in matching. In Dean Thomas, editor, *Proceedings of IJCAI-99 (Vol2)*, pages 1198–1203, S. F., July 31–August 6 1999. Morgan Kaufmann Publishers. 10

[5] G. Gottlob and A. Leitsch. On the efficiency of subsumption algorithms. *Journal of the Association for Computing Machinery*, 32(2):280–295, 1985. 2

[6] N. Helft. Inductive generalization: A logical framework. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning*, pages 149–157, Wilmslow, UK, 1987. Sigma Press. 9

[7] J.-U. Kietz and M. Lübbe. An efficient subsumption algorithm for inductive logic programming. In W. Cohen and H. Hirsh, editors, *Proceedings of ICML-94*, pages 130–138, 1994. 2

[8] J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, New York, 2 edition, 1987. 1

[9] J. Maloberti and M. Sebag. $\theta$-subsumption in a constraint satisfaction perspective. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of ILP 2001*, volume 2157 of *Lecture Notes in Artificial Intelligence*, pages 164–178. Springer, September 2001. 3, 9, 10

[10] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–49, January 1965. 1

[11] T. Scheffer, R. Herbrich, and F. Wysotzki. Efficient $\theta$-subsumption based on graph algorithms. In Stephen Muggleton, editor, *Proceedings of ILP-96*, volume 1314 of *LNAI*, pages 212–228. Springer, August 26–28 1997. 2

[12] M. Schmidt-Schauss. Implication of clauses is undecidable. *Theoretical Computer Science*, 59:287–296, 1988. 1

[13] Ashwin Srinivasan, Stephen Muggleton, Michael J. E. Sternberg, and Ross D. King. Theories for mutagenicity: A study in first-order and feature-based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996. 9

[14] R. B. Stillman. The concept of weak substitution in theorem-proving. *Journal of ACM*, 20(4):648–667, October 1973. 2

# Temporal Decision Trees for Diagnosis: An Extension

Claudia Picardi

Dipartimento di Informatica
Università degli Studi di Torino
Corso Svizzera 185, 10149, Torino, Italy.
claudia.picardi@di.unito.it

**Abstract.** Model-Based Diagnosis often cannot be directly exploited for embedded software, which must run with very strict constraints on memory and time. It is however possible to compile the knowledge explicited by a model-based diagnostic engine into a decision tree that serves as the basis for on-board diagnostic software. In order to exploit temporal information possibly present in the model, temporal decision trees for diagnosis have been introduced [5, 6]. This paper presents an extension to the temporal decision trees framework that widens its applicability.

## 1 Introduction

Electronic components are nowadays embedded in many devices, ranging from large-scale low-cost productions such as cars to much more expensive and sophysticate systems such as aircrafts or spacecrafts. Devices are thus controlled essentially by on-board software, which runs on Electronic Control Units (ECUs) and takes care of most aspects of the system's behaviour. A significant part of on-board software is devoted to diagnosis: the system must be able to react to failures by performing appropriate recovery actions [9], in order to avoid further damage or to restore lost functionality.

Model-based diagnosis, which automates the diagnostic process by reasoning on a model of the diagnosed system, has repeatedly proved itself as a valuable tool for off-board diagnostics (see for example [3, 12, 17], or [4] for a more general discussion); however the restrictions thet embedded software must comply with make it less amenable to on-board use. This is especially true for low-cost products, where the number and size of ECUs strongly limits the amount of time and memory that the diagnostic software can exploit.

Proposed solutions (to cite some, [2, 7, 18]) mostly consist in running a model-based software off-line in order to build a table matching observed data with faults and recovery actions, and to induce from such table a set of rules that can be the basis for a much more compact and swift diagnostic software. [2] in particular proposes to build decision trees, which have been exploited also for diagnostic applications different than on-board software (see [8, 11, 15]).

A main limitation of this technique is that, while model-based diagnosis is being extended to cope with dynamic systems and temporal constraints, decision

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 14–26, 2003.
© Springer-Verlag Berlin Heidelberg 2003

trees are not able to deal with temporal information. This results in a loss of diagnostic capability when passing from a full-fledged model based engine to the decision tree compiled from it. For this reason [5, 6] propose to exploit *temporal decision trees*, an extension of decision trees that takes into account also temporal information. These papers develop an algorithm (TID3, an extension of the ID3 algorithm propsed by Quinlan [16]) that builds temporal decision trees from a set of temporal examples, considering a number of constraints, such as the severity of the selected recovery actions, the deadline by which a recovery action should be performed, and last but not least the depth of the resulting tree.

Unfortunately, the algorithm proposed in the above works suffers from a few limitations, which make it not suitable to some diagnostic scenarios. The first regards recovery actions: TID3 takes into account the severity of recovery actions, but assumes that this parameter cannot change depending on the particular fault. This is often not true: a recovery action can have serious drawbacks if performed in presence of a fault different than the one the action has been introduced for. This may happen when the on-board software is not able to discriminate among multiple candidate faults: in this case it performs a combination of all corresponding recovery actions.

The second limitation regards deadlines: TID3 assumes that there is a *hard* deadline, that is, that the recovery action must be selected within a given time interval. This is not a very general approach; in a sense it assumes that there is no cost in waiting, while exceeding the deadline has an infinite cost. It does not take into account that there can be a trade-off between waiting beyond the deadline and the resulting gain in information.

For this reason, in this paper we propose two weakenings of the assumptions behind TID3, which result in an extension of the scenario where the algorithm can be applied. The extension takes into account *recovery actions with drawbacks* and *soft deadlines*, that is deadlines expressed as an increase in cost as time passes.

The paper is organized as follows: section 2 introduces the notion of temporal decision tree, and the original diagnostic scenario TID3 was developed for. Section 3 discusses the proposed extensions, while section 4 shows how TID3 can be modified, without altering its performance, in order to deal with the new scenario. Section 5 draws the conclusions outlining future work and comparing the proposed technique with related research.

## 2 Building Temporal Decision Trees

A diagnostic software based on decision trees exploits a pre-compiled tree in order to know which sensors it should read and which rcovery action it should select.

The software traverses the tree on a path from the root to a leaf: inner nodes are labelled with sensor names and they tell the software the next sensor to read.

Outgoing edges are labelled with sensor values[1], thus according to the value it reads the software selects one of the node's children. Each leaf is labelled with a recovery action that the software performs when the leaf is reached.

When the decision tree is *temporal*, inner nodes are labelled not only with a sensor name, but also with a time label. When the software starts traversing the tree, it activates an internal clock (thus time 0 corresponds to the instant of fault detection). The software reads the value of a sensor suggested by an inner node only when its internal clock matches the corresponding time label. It is obvious that time labels must not decrease along a path from the root to a leaf.

For every node $n$, we will denote by $\mathcal{L}(n)$ its label and by $\mathcal{T}(n)$ its time label. Leaves do not have proper time labels, since after a recovery action has been selected it does not make sense to wait before performing it. For a tree leaf $l$, $\mathcal{T}(l)$ thus represents the time at which the action is performed, and it coincides with the time label of the parent node. For any edge $\langle n, c\rangle$ we will denote by $\mathcal{L}(\langle n, c\rangle)$ the value of sensor $\mathcal{L}(n)$ that leads to $c$.

In order to automatically generate a temporal decision tree, one has first of all to define the *diagnostic scenario*, which is characterized by:

1. the available sensors;
2. the possible faults;
3. for each fault, the corresponding recovery action;
4. for each fault, the deadline by which the recovery action must be performed;
5. a model for recovery actions $\mathbf{A}$, which gives more structured information about available actions. In particular, $\mathbf{A}$ defines
   (a) the set of all available actions;
   (b) a partial ordering $\prec$ on recovery actions that expresses recover capability: $a_1 \prec a_2$ means that $a_2$ is *stronger* than $a_1$ and can thus be performed in its place. This information is used when the decision tree is not able to discriminate two faults with two different recovery actions: in this case the software must in fact select the *weakest* recovery action which is *stronger* than both recovery actions.
   (c) a cost $\chi(a)$ for each recovery action $a$, which must be increasing with respect to $\prec$. The cost expresses numerically the severity of the recovery action, and it quantifies the loss when an action $a_1$ is substituted for a stronger action $a_2$.

TID3 builds a temporal decision tree starting from temporal set of examples (te-set for short) for the scenario of interest. The te-set, denoted by $\mathbb{E}$, describes a set of *fault situations*; for each of them it reports the values sensors show over time, the probability and, depending on the fault, the recovery action that should be performed, along with the corresponding deadline. With each te-set $\mathbb{E}$ we can associate a set of time labels $t_1, \ldots, t_l$, corresponding to the times at which sensor values are collected. Of course not all fault situations need to contain the sensor values for every time label.

[1] We consider qualitative values, as it happens in Model-Based Diagnosis.

The purpose of TID3 is not only to build a reasonably small decision tree, but also to to deal with the additional constraints imposed by the fact that the tree is temporal. Some of these constraints are *hard*, meaning that either they are met (and the resulting tree is acceptable) or not. For example, the time label of a node must not be greater than the time labels of its children: violating this requirement produces an invalid decision tree.

Some other constraints are *soft*, since they can be met with various degrees of satisfaction. An example of this is the selection of recovery actions: selecting a recovery action which is stronger than needed is possible, but not desirable, because the cost is probably higher. [5, 6] introduce a notion of *cost* of a tree, the $\mathcal{X}$-cost, that measures the degree of satisfaction of soft constraints.

Given a temporal decision tree $\mathbf{T}$ built over a te-set $\mathbb{E}$ and a model for recovery actions $\mathbf{A}$, the $\mathcal{X}$-cost of its nodes is inductively defined as follows:

$$\mathcal{X}_{\mathbb{E},\mathbf{A}}(n) = \begin{cases} \chi(\mathcal{L}(n)) & \text{if } n \text{ is a leaf} \\ \sum\limits_{c \text{ child of } n} P(\mathbb{E}(c)|\mathbb{E}(n)) \cdot \mathcal{X}_{\mathbb{E},\mathbf{A}}(c) & \text{if } n \text{ is an inner node} \end{cases}$$

The $\mathcal{X}$-cost of $\mathbf{T}$ coincides with the $\mathcal{X}$-cost of its root.

$\mathcal{X}$-costs depend on the te-set $\mathbb{E}$ and the action model $\mathbf{A}$ used for building the tree: the latter provides the cost of the actions corresponding to the leaves, while the former provides the probabilities of the fault situations. In particular, in the definition of $\mathcal{X}$-cost, we see the expression $P(\mathbb{E}(c)|\mathbb{E}(n))$. Given a node $n$, $\mathbb{E}(n)$ denotes those fault situations in $\mathbb{E}$ that in traversing the tree would lead through $n$. The probability $P(\mathbb{E}(n))$ is the sum of the individual probabilities of those fault situations, and is inductively considered as the probability that $n$ is visited. Analogously, $P(\mathbb{E}(c)|\mathbb{E}(n))$ is defined as $P(\mathbb{E}(c))/P(\mathbb{E}(n))$, and it represents the conditional probability that a fault situation leading through $n$ leads also through its child $c$.

TID3 is able to build a tree with minimum $\mathcal{X}$-cost, while at the same time exploiting entropy as ID3 does in order to keep the tree small. Moreover, the asymptotical complexity of TID3 is proven to be the same as that of ID3 itself, that is $O(N^2MT)$ in the worst case, and $O(NMT \log N)$ in the best case, where $N$ denotes the number of examples, $M$ the number of sensors, and $T$ the number of time labels.

## 3 The Extended Scenario

Both the extensions we propose — namely, action drawbacks and soft deadlines — require a change in the notion of diagnostic scenario, and a consequent redefinition of the tree cost. Let us start with considering action drawbacks.

**Action Drawbacks** occur when a recovery action is performed in a different context than the one it is suited for. We have seen how, when the software is not able to discriminate two fault situations $\mathbf{e}_1, \mathbf{e}_2$, it may need to perform a combination of their recovery actions. It can happen that such combination has a different cost depending on whether the actual situation is $\mathbf{e}_1$ or $\mathbf{e}_2$. This

can be expressed by saying that the cost of recovery actions *depends on the fault situation they are performed in.*

The model for recovery actions still defines the set of available actions, the partial ordering $\prec$ and the individual action cost; but the action cost, which was a single value $\chi(a)$, becomes now a function $\gamma(a, \cdot) : \mathbb{E} \mapsto \mathbb{R}^+$. We also need a monotonicty condition, stating that in each fault situation $\mathbf{e}$ $\gamma(\cdot, \mathbf{e})$ must be increasing wrt $\prec$.

Consequently, we have a new notion of tree cost — called $\mathcal{Y}$-cost — that takes into account this extended recovery action model:

**Definition 1** Let $\mathbf{T}$ be a temporal decision tree built over a te-set $\mathbb{E}$ and a model for recovery actions $\mathbf{A}$. The $\mathcal{Y}$-cost of a tree node $n$ is then inductively defined as follows:

$$\mathcal{Y}_{\mathbb{E},\mathbf{A}}(n) = \begin{cases} \sum\limits_{\mathbf{e}\in\mathbb{E}(n)} P(\mathbf{e}|\mathbb{E}(n)) \cdot \gamma(\mathcal{L}(n), \mathbf{e}) & \text{if } n \text{ is a leaf} \\ \sum\limits_{c \text{ child of } n} P(\mathbb{E}(c)|\mathbb{E}(n)) \cdot \mathcal{Y}_{\mathbb{E},\mathbf{A}}(c) & \text{if } n \text{ is an inner node} \end{cases} \tag{1}$$

The $\mathcal{Y}$-cost of a temporal decision tree is then the $\mathcal{Y}$-cost of its root. $\diamond$

Notice that $\mathcal{Y}$-cost is equal to $\mathcal{X}$-cost whenever action costs do not change across different fault situations. Thus $\mathcal{Y}$-cost is a proper extension of $\mathcal{X}$-cost. Moreover, $\mathcal{Y}$-cost shares with $\mathcal{X}$-cost the following two properties[2]:

***Proposition 2*** *The $\mathcal{Y}$-cost of a temporal decision tree $\mathbf{T}$ depends only on its leaves; more precisely:*

$$\mathcal{Y}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) = \sum_{l \text{ leaf of } \mathbf{T}} \left( \sum_{\mathbf{e}\in\mathbb{E}(l)} P(\mathbf{e}|\mathbb{E}) \cdot \gamma(\mathcal{L}(l), \mathbf{e}) \right) \tag{2}$$

***Proposition 3*** *Let $\mathbf{T}, \mathbf{U}$ denote two temporal decision trees, built over the same te-set $\mathbb{E}$ and the same action model $\mathbf{A}$. Suppose that $\mathbf{T}$ is* more discriminating *than $\mathbf{U}$, that is, for each fault situation $\mathbf{e}$, $\mathbf{T}$ associates to $\mathbf{e}$ a less or equally expensive action than $\mathbf{U}$, and there exists at least one fault situation $\mathbf{e}$ for which the recovery action selected by $\mathbf{T}$ is actually less expensive than the one selected by $\mathbf{U}$. Then $\mathcal{Y}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) < \mathcal{Y}_{\mathbb{E},\mathbf{A}}(\mathbf{U})$.* $\diamond$

**Soft Deadlines** instead represent the additional cost due to the posticipation of the recovery action. The longer the delay, the higher the cost. However, the information acquired in waiting could allow to select a cheaper recovery action, thus there is a trade-off between the two choices.

Let us consider a te-set $\mathbb{E}$ with time labels $t_1 < \ldots < t_l$. A soft deadline can be represented as a function $\delta : \mathbb{E} \times \{t_1, \ldots, t_l\} \mapsto \mathbb{R}^+$ where $\delta(\mathbf{e}, t_i)$ represents

[2] This paper does not contain proofs for the sake of conciseness. The interested reader can find an extended version in the technical report [13].

the cost of performing a recovery action for fault situation $\mathbf{e}$ at time $t_i$. $\delta$ must satisfy the following requirement: for every $\mathbf{e} \in \mathbb{E}$, and for every pair of time labels $t_i < t_j$, it must hold that $\delta(\mathbf{e}, t_i) \leq \delta(\mathbf{e}, t_j)$.

In this case the expected cost of a decision tree must take into account not only which recovery action is performed for a given fault situation, but also when it is performed. We thus define a new notion of expected cost, which we will call $\mathcal{W}$-cost:

**Definition 4** Let $\mathbf{T}$ be a temporal decision tree built over a te-set $\mathbb{E}$ and a model for recovery actions $\mathbf{A}$.

The time label function $\mathcal{T}$ is defined only on inner nodes; we extend it to tree leaves by saying that the time label of a tree leaf is the same as the one of its parent node. In this way the time label of a tree leaf denotes the time at which the recovery action is performed.

The $\mathcal{W}$-cost of a tree node $n$ is then inductively defined as follows:

$$\mathcal{W}_{\mathbb{E},\mathbf{A}}(n) = \begin{cases} \sum\limits_{\mathbf{e}\in\mathbb{E}(n)} P(\mathbf{e}|\mathbb{E}(n)) \cdot (\gamma(\mathcal{L}(n), \mathbf{e}) + \delta(\mathbf{e}, \mathcal{T}(n))) & \text{if } n \text{ is a leaf} \\ \sum\limits_{c \text{ child of } n} P(\mathbb{E}(c)|\mathbb{E}(n)) \cdot \mathcal{W}_{\mathbb{E},\mathbf{A}}(c) & \text{if } n \text{ is an inner node} \end{cases} \tag{3}$$

The $\mathcal{W}$-cost of a temporal decision tree is, as usual, the $\mathcal{W}$-cost of its root. $\diamond$

In this case in order to fall back in the case of hard deadlines we simply have to define properly the function $\delta$, so that $\delta(\mathbf{e}, t_i)$ is 0 whenever $t_i$ is within the hard deadline, while for $t_i$s beyond the deadline $\delta(\mathbf{e}, t_i)$ is so high that $t_i$ is never worth waiting for[3].

The following proposition holds:

***Proposition 5*** *Let $\mathbf{T}$ be a temporal decision tree built over a te-set $\mathbb{E}$ and $n$ action model for $\mathbf{A}$. Then the $\mathcal{W}$-cost of $\mathbf{T}$ can be expressed as:*

$$\mathcal{W}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) = \mathcal{Y}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) + \sum_{\mathbf{e}\in\mathbb{E}} P(\mathbf{e}|\mathbb{E}) \cdot \delta(\mathbf{e}, \mathcal{T}(\mathbf{leaf_T}(\mathbf{e}))) \tag{4}$$

*where $\mathbf{leaf_T}(\mathbf{e})$ denotes the tree leaf $l$ such that $\mathbf{e} \in \mathbb{E}(l)$.*

This tells us that the $\mathcal{W}$-cost of a tree can be computed as the sum of its $\mathcal{Y}$-cost, expressing the cost due to the selection of recovery actions, and a term expressing the cost due to elapsed time. In other words, the two contributions to the cost due to discriminating capability and delay in performing recovery actions, can be separated. Lowering one of the two can mean increasing the other, so the problem is to find a trade-off between them.

[3] It is always possible to find a high enough number, by adding 1 to the cost of the most expensive recovery action in $\mathbb{A}$. In this case in fact the advantage obtained thanks to the new information is surely lower than the loss due to the cost of obtaining the information (i.e. the cost of waiting until $t_i$).

## 4 Temporal Decision Trees for the Extended Scenario

Changing the cost function — as we did in the previous section — can have an insignificant or enormous impact on the generation of decision trees, depending on whether the cost function satisfies or not certain properties.

The correctness proof of TID3[4], which can be found in [6], is based on two assumptions, which are necessary and sufficient conditions for it to hold:

1. the cost function must depend only on the way the fault situations split among the tree leaves, and not on the internal structure of the tree;
2. the cost function must correctly express the discriminating power of a tree: a more discriminating tree should have a lower cost.

Propositions 2 and 3 tell us that $\mathcal{Y}$ satisfies the assumptions. Thus in order to update the algorithm we only need to modify the subroutine computing costs so that it computes $\mathcal{Y}$-costs instead of $\mathcal{X}$-costs. The two cost functions are very similar: both consider all fault situations associated with the leaf. For $\mathcal{X}$-costs one only has to determine the recovery action and its individual cost, while for $\mathcal{Y}$-costs one has to sum up the weighted costs of the recovery action in all different fault situations. In other words, where the former cycles once on the set of associated fault situations, the latter cycles twice. This modification does not change the asymptotical complexity of TID3.

Now we need to consider $\mathcal{W}$-costs: unfortunately, it is easy to see that $\mathcal{W}$ does not satisfy any of the two conditions required for the algorithm to work. First of all, two trees with the same leaves (i.e. that associate the same recovery actions to the same situations) can have different costs, since the cost depends also on the time at which recovery actions are performed. Secondly, a more discriminating tree can have a higher cost than a less discriminating one, since the former may perform later its recovery actions.

Thus we cannot directly apply TID3 to problems that deal with soft deadlines. Even if the algorithm cannot be used as it is, however, it can still be exploited to generate temporal decision trees. In fact, in the following we will show that the problem of generating a temporal decision tree with soft deadlines is reducible to the problem with hard deadlines, by running a pre-processing step that does not alter the asymptotical complexity of the overall algorithm.

The first step consists in computing the minimum possible $\mathcal{W}$-cost for a given te-set. Figures 1 and 2 show the proposed algorithm for doing this. The idea is the following: first we build an *exhaustive tree* (line 4), that is a temporal decision tree that exploits *all* available observations in a fixed order. This means that for any two inner nodes $n_1, n_2$ that are at the same level $\mathcal{L}(n_1) = \mathcal{L}(n_2)$ and $\mathcal{T}(n_1) = \mathcal{T}(n_2)$. We choose arbitrarily the order in which the observations must occur along a path from the root to a leaf: they must be ordered by time label (otherwise we would not get a proper temporal decision tree) and then by

[4] That is, the proof that it produces a tree with the minimum possible $\mathcal{X}$-cost for the initial te-set.

sensor index. Thus we have the following sequence:

$$\langle s_1, t_1\rangle, \langle s_2, t_1\rangle, \ldots, \langle s_m, t_1\rangle, \langle s_1, t_2\rangle, \ldots, \langle s_m, t_2\rangle, \ldots, \langle s_1, t_l\rangle, \ldots, \langle s_m, t_l\rangle$$

We do not provide a detailed description of BuildExhaustiveTree since it is quite trivial to implement it.

After building the exhaustive tree `T`, we *prune it* (line 6) in order to get a pruned tree `T_min` whose cost `min_wcost` is also the minimum possible $\mathcal{W}$-cost for any temporal decision tree built over the te-set and the action model in input (respectively denoted in the algorithm by `Ex` and `Act`).

In general pruning a tree means removing some of its subtrees, thus transforming some inner nodes in leaves. For a temporal decision tree this means not only removing all the children nodes, but also changing the label so that it corresponds to the proprer recovery action. For this reason the PruneTree function takes in input also the te-set and the action model.

There are many different prunings of the same tree; function PruneTree (figure 2) builds a specific pruning with the following properties: *(i)* the resulting tree has the minimum possible $\mathcal{W}$-cost with respect to `Ex` and `Act`; *(ii)* the pruning carried out is the minumum necessary in order to have a tree with minimum cost.

Let us examine PruneTree in more detail. Lines 13–18 deal with those cases where the input tree is made of a single leaf. Since a leaf cannot be further pruned, the returned tree `U` is identical to the input tree `T`; its $\mathcal{W}$-cost is computed by function WCost according to definition 4.

If the input tree is not a leaf, then there are two ways to prune it: either pruning its subtrees or transforming the root in a single leaf. Lines 19–28 follow the first approach: the input te-set is split according to the tree root (line 23) and PruneTree is recursively called on the subtrees (line 25). We create a new tree `U1` that has the same root as `T`, and has as subtrees the prunings of `T`'s subtrees. Its $\mathcal{W}$-cost is stored in `wcost1`, and it is the minimum $\mathcal{W}$-cost among all prunings *where the root is not a transformed in a leaf.*

Lines 30–34 consider this last option: `U2` represents the pruning where the root becomes a leaf, and `wcost2` is its $\mathcal{W}$-cost. At this point we only have to choose between `U1` and `U2` the one with minimum cost, and return it. In case

```
1  function GetMinWCost(te-set Ex, action model Act)
2  returns min_wcost, the minimum possible W-cost for Ex and Act.
3  begin
4      T ← BuildExhaustiveTree(Ex, Act);
5      tstart ← smallest time label in Ex;
6      ⟨T_min, min_wcost⟩ ← PruneTree(T, tstart, Ex, Act);
7      return min_wcost;
8  end.
```

**Fig. 1.** Algorithm for computing the minimum possible $\mathcal{W}$-cost for a te-set in input

```
9   function PruneTree (tree T = ⟨root, Nodes, Edges, Labels, TLabels⟩,
                                time label tparent, te-set Ex, action model Act)
10  returns a pair ⟨U, wcost⟩ where U is the pruned tree, wcost is its W-cost.
11  begin
12      out ← {⟨n, c⟩ | root = n};
13      if out is empty then begin      {in this case root is a leaf}
14          U_Labels(root) ← Labels(root); U_TLabels(root) ← tparent;
15          U ← ⟨root, {root}, ∅, U_Labels, U_TLabels⟩;
16          wcost ← WCost(U, Ex, Act);
17          return ⟨U, wcost⟩;
18      end;
19      U1_Labels(root) ← Labels(root); U1_TLabels(root) ← TLabels(root);
20      U1 ← ⟨root, {root}, ∅, U1_Labels, U1_TLabels⟩;
21      wcost1 ← 0;
22      for all ⟨root, child⟩ in out do begin
23          SubEx ← {e ∈ Ex | value of observation Labels(root) for e
                                    is equal to Labels(⟨root, child⟩)};
24          SubT ← ⟨child, Nodes, Edges, Labels, TLabels⟩;
25          ⟨SubU, subcost⟩ ← PruneTree(SubT, TLabels(root), SubEx, Act);
26          wcost1 ← wcost1 + subcost ∗ Prob(SubEx, Ex);
27          Append(U1, root, SubU);
28      end;
29      if tparent > TLabels(root) then begin
30          U2_Labels(root) ← merge of all actions for all e ∈ Ex;
31          U2_TLabels(root) ← tparent;
32          U2 ← ⟨root, {root}, ∅, U2_Labels, U2_TLabels⟩;
33          wcost2 ← WCost(U2, Ex, Act);
34          if wcost2 < wcost1 then return ⟨U2, wcost2⟩;
35      end;
36      return ⟨U1, wcost1⟩;
37  end.
```

**Fig. 2.** Algorithm for pruning the exhaustive tree and obtaining the minimum $\mathcal{W}$-cost

they have the same cost, we prefer to return the "less invasive" pruning, that is U1, where fewer nodes are cut.

Notice (line 29) that U2 is considered only if the time label of the tree root is higher than that of its parent (recall that what the function considers the root could be an inner node, since the function is recursively called on subtrees). In fact, if the parent has the same time label as the current node, $\mathcal{W}$-cost would not be lowered by transforming it in a leaf.

We have the following theorem:

***Theorem 6*** *Let us consider a call to* GetMinWCost *with a te-set* $\mathbb{E}$ *and an action model* $\mathbf{A}$ *as inputs. Let us consider the tree* T_min *built on line 6. Then the following statements hold:*

```
1   function ExtendedTID3(te-set Ex, action model Act)
2   returns V, the tree with minimum possible W-cost for Ex and Act.
3   begin
4       T ← BuildExhaustiveTree(Ex, Act);
5       tstart ← smallest time label in Ex;
6       ⟨T_min, min_wcost⟩ ← PruneTree(T, tstart, Ex, Act);
7       HardDl ← ∅;    {HardDl is the hard deadlines function}
8       TLabels ← time labelling function of T_min;
9       for all leaves l of T_min do
10          for all fault situations e ∈ Ex(l) do
11              HardDl(e) ← TLabels(l);
12      Ex_new ← Ex with hard deadlines as in HardDl;
13      Obs ← all observations ⟨s, t⟩ in Ex_new;
14      V ← TID3(Ex_new, Act);
15      return V;
16  end.
```

**Fig. 3.** Algorithm for building a tree with minimum $\mathcal{W}$-cost

1. *For every temporal decision tree* $\mathbf{T}$ *built over* $\mathbb{E}$ *and* $\mathbf{A}$, $\mathcal{W}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) \geq \mathcal{W}_{\mathbb{E},\mathbf{A}}(\texttt{T_min})$. *Thus* `min_wcost` *effectively contains the minimum possible* $\mathcal{W}$*-cost for* $\mathbb{E}, \mathbf{A}$.
2. *For every temporal decision tree* $\mathbf{T}$ *built over* $\mathbb{E}$ *and* $\mathbf{A}$, *if there exists* $e \in \mathbb{E}$ *such that* $\mathcal{T}(\mathbf{leaf_T}(e)) > \mathcal{T}(\mathbf{leaf}_{\texttt{T_min}}(e))$ *then* $\mathcal{W}_{\mathbb{E},\mathbf{A}}(\mathbf{T}) > \mathcal{W}_{\mathbb{E},\mathbf{A}}(\texttt{T_min})$. $\diamond$

Now let us see how this can help us in exploiting TID3. TID3 builds a temporal decision tree with minimum $\mathcal{X}$-cost starting from a te-set with hard deadlines. What we need to do is to build a temporal decision tree with minimum $\mathcal{W}$-cost starting from a te-set with soft deadlines.

What we will show is that for every te-set $\mathbb{E}$ with soft deadlines it is possible to build a te-set $\mathbb{E}^*$ with hard deadlines such that a temporal decision tree with minimum $\mathcal{X}$-cost built over $\mathbb{E}^*$ has minimum $\mathcal{W}$-cost wrt $\mathbb{E}$.

$\mathbb{E}^*$ can be obtained from $\mathbb{E}$ by simply associating to each fault situation $e \in \mathbb{E}$ a hard deadline instead of a soft one. Such a hard deadline is given from the pruned tree `T_min`: more precisely, *the hard deadline* $\mathbf{Dl}(e)$ *associated with a fault situation* $\mathbf{e}$ *is given by the time label associated by* `T_min` *to the leaf corresponding to* $e$*: in other words* $\mathbf{Dl}(e) = \mathcal{T}_{\texttt{T_min}}(\mathbf{leaf}_{\texttt{T_min}}(e))$. This choice of hard deadlines prevents TID3 from trying to exploit information that would necessarily lead to a tree with a non-minimum $\mathcal{W}$-cost, as stated by theorem 6.2. Thus the information we exclude by imposing hard deadlines is such that it could not be used in any case in a temporal decision tree with minimum $\mathcal{W}$-cost.

Figure 3 shows how this approach can be implemented in order to get a temporal decision tree with minimum $\mathcal{W}$-cost. First of all we build the pruned tree, in exactly the same way as we did in figure 1. After this, we use the time labels of the leaves in the pruned tree in order to obtain hard deadlines (lines 7–11), represented by variable `HardDl`. Then we build a modified te-set that uses hard

deadlines instead of soft ones (line 12), and we call TID3 on it (line 14) in order to obtained the desired tree.

The correctness of this approach is proven by the following theorem:

***Theorem 7*** *Let us consider a call to* EXTENDEDTID3 *with a te-set* $\mathbb{E}$ *and a recovery action model* $\mathbf{A}$ *as input. The returned tree* $\mathtt{V}$ *is such that* $\mathcal{W}_{\mathbb{E},\mathbf{A}}(\mathtt{V}) = \mathcal{W}_{\mathbb{E},\mathbf{A}}(\mathtt{T_min})$.

As to complexity, the additional overhead of EXTENDEDTID3 is that of building the exhaustive tree and pruning it. These two operations require $O(NMT)$, that is they are linear in the size of the examples table. This is due to the fact that the exhaustive tree is essentially a reformulation of such table.

Since, as we pointed out at the end of section 2, the complexity of TID3 is $O(N^2MT)$ in the worst case and $O(NMT \log N)$ in the best case, the overall asymptotical complexity remains unchanged.

## 5 Conclusions

In this paper we have extended the algorithm introduced in [5, 6] which generates temporal decision trees for diagnosis. The extension we introduced allows to apply this technique in a wider range of scenarios; in particular the possibility to express soft deadlines provides a useful generalization of the original approach, which made the assumption that "waiting has no cost".

The new approach presented in this paper allows instead to take into consideration the trade-off between a loss of time and a loss of information.

As pointed out in [6], in the diagnostic literature the notion of temporal decision tree is new, while there exists some related work in the field of fault prevention. In particular, [10] proposes to build temporal decision trees in order to extract from time series of sensor readings the relevant information which allows to predict a fault occurrence.

The algorithm proposed by [10] has some substantial differences both in the goal (for example, it performs only binary classification) and in the architecture. TID3 and EXTENDEDTID3 work greedily from the point of view of the tree size, using entropy as heuristic, and optimize the cost of the tree; on the other hand, the algorithm from [10] works greedily wrt the tree cost, and then tries to reduce the tree size with pruning.

The evaluation of the tree cost proposed in [10] takes into account the trade-off between discrimination capability and delay in the final decision, thus being more similar to $\mathcal{W}$-costs than to $\mathcal{X}$-costs. The extension we have proposed in this paper would therefore allow to apply our technique also to the kind of problems [10] deals with.

Future work on this topic aims mainly at considering its applicability in another area where decision trees are used: that of *data mining*. An investigation of existing papers (see [1] for an overview) seems to suggest that, whereas the analysis of temporal sequences of data has received much interest in the last

years, not much work has been done in the direction of data classification, where temporal decision trees could be exploited. Also with this respect, we believe that the extension proposed in this paper can help in generalizing the technique to other areas.

## References


[1] C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. In *KDD Workshop on Temporal Data Mining*, San Francisco, August 2001. 24

[2] F. Cascio, L. Console, M. Guagliumi, M. Osella, A. Panati, S. Sottano, and D. Theseider Dupré. Generating on-board diagnostics of dynamic automotive systems based on qualitative deviations. *AI Communications*, 12(1):33–44, 1999. 14

[3] F. Cascio and M. Sanseverino. IDEA (Integrated Diagnostic Expert Assistant) model-based diagnosis in the the car repair centers. *IEEE Expert*, 12(6), 1997. 14

[4] L. Console and O. Dressler. Model-based diagnosis in the real world: lessons learned and challenges remaining. In *Proc. 16th IJCAI*, pages 1393–1400, Stockholm, 1999. 14

[5] L. Console, C. Picardi, and D. Theseider Dupré. Temporal Decision Trees or the lazy ECU vindicated. In *Proc. of the 17th IJCAI*, volume 1, pages 545–550, 2001. 14, 15, 17, 24

[6] L. Console, C. Picardi, and D. Theseider Dupré. Temporal decision trees: Bringing model-based diagnosis on board. *Journal of Artificial Intelligence Research*, 2003. To appear. Also in [14]. 14, 15, 17, 20, 24

[7] A. Darwiche. On compiling system descriptions into diagnostic rules. In *Proc. 10th Int. Work. on Principles of Diagnosis*, pages 59–67, 1999. 14

[8] P.-P. Faure, L. Travé Massuyès, and H. Poulard. An interval model-based approach for optimal diagnosis tree generation. In *Proceedings of the 10th International Workshop on Principles of Diagnosis (DX-99)*, 1999. 14

[9] G. Friedrich, G. Gottlob, and W. Nejdl. Formalizing the repair process. In *Proc. 10th ECAI*, pages 709–713, Vienna, 1992. 14

[10] P. Geurts and L. Wehenkel. Early prediction of electric power system blackouts by temporal machine learning. In *Proceedings of the ICML98/AAAI98 Workshop on "Predicting the future: AI Approaches to time series analysis"*, Madison, July 24-26, 1998. 24

[11] H. Milde and L. Lotz. Facing diagnosis reality - model-based fault tree generation in industrial application. In *Proceedings of the 11th International Workshop on Principles of Diagnosis (DX-00)*, pages 147–154, 2000. 14

[12] P. J. Mosterman, G. Biswas, and E. Manders. A comprehensive framework for model based diagnosis. In *Proc. 9th Int. Work. on Principles of Diagnosis*, pages 86–93, 1998. 14

[13] C. Picardi. How to exploit temporal decision trees in case of action drawbacks and soft deadlines. Technical Report, March 2003. Downloadable at: http://www.di.unito.it/%7epicardi/Download/Papers/tr_picardi010303.ps.gz. 18

[14] C. Picardi. *Diagnosis: From System Modelling to On-Board Software*. PhD thesis, Dipartimento di Informatica, Università di Torino, Downloadable at http://www.di.unito.it/%7epicardi/, 2003. 25

[15] C. J. Price. AutoSteve: Automated electrical design analysis. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI2000)*, pages 721–725, 2000. 14

[16] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 15

[17] M. Sachenbacher, A. Malik, and P. Struss. From electrics to emissions: experiences in applying model-based diagnosis to real problems in real cars. In *Proc. 9th Int. Work. on Principles of Diagnosis*, pages 246–253, 1998. 14

[18] M. Sachenbacher, P. Struss, and R. Weber. Advances in design and implementation of obd functions for diesel injection systems based on a qualitative approach to diagnosis. In *SAE 2000 World Congress*, 2000. 14

# Obligations as Social Constructs

Guido Boella and Leendert van der Torre

[1] Dipartimento di Informatica – Università di Torino – Italy
guido@di.unito.it
[2] SEN-3 – CWI Amsterdam – The Netherlands
torre@cwi.nl

**Abstract.** In this paper we formalize sanction-based obligations in the context of Searle's notion of construction of social reality. In particular, we define obligations using a *counts as* conditional, Anderson's reduction to alethic modal logic and Boella and Lesmo's normative agent. Our analysis presents an alternative criticism to the weakening rule, which has already been criticized in the philosophical literature for its role in the Ross paradox and the Forrester paradox, and the analysis presents a criticism to the generally accepted conjunction rule. Moreover, we show a possible application of these results in a qualitative decision theory. Finally, our analysis also contributes to philosophical discussions such as the distinction between violations and sanctions in Anderson's reduction, and between implicit and explicit normative systems.

## 1 Introduction

In agent theory, mental and social attitudes used in folk psychology such as knowledge, beliefs, desires, goals, intentions, commitments, norms, obligations, permissions, *et cetera*, are attributed to artificial systems [15]. The conceptual and logical study of these attitudes changes with the change of emphasis from autonomous agent systems to multiagent systems. For example, new challenges have been posed by new forms of multiagent systems such as web based virtual communities realized by the grid and peer to peer paradigms. In these settings it is not possible to design a central control since they are made of heterogeneous agents which cannot be assumed always to stick to the system regulations. The main driving force of single agent systems was Newell and Simon's study of knowledge and goals as knowledge level concepts in bounded or limited reasoning in knowledge based systems [20], and more recently Bratman's study of intentions as, amongst others, stabilizers of behavior in the agent's deliberation and planning process [9]. Likewise, joint intentions, joint commitments, norms and obligations are studied as stabilizers of multiagent systems.

However, from philosophical and sociological studies it is well known that there is more to multiagent concepts than stabilizing behavior. For example, multiagent behavior may spontaneously emerge without being reducible to the behavior of individual agents (known as the micro-macro dichotomy). Moreover, in a society the emersion of normative concepts is possible since they are constructed due to social processes. Searle's notion of construction of social reality

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 27–39, 2003.
© Springer-Verlag Berlin Heidelberg 2003

explains these processes, e.g., how due to social conventions banknotes may be more than just pieces of paper, and what it means to be married [22]. The core concept of this construction is that in a social reality, certain actions and facts may *count as* something else. Under certain conditions, a priest performing a ritual counts as marrying a couple. Considering normative conceptions inspired by how human societies work and are constructed may have a decisive role also in the coordination of multiagent systems such as virtual communities, especially when artificial agents have to interact with human ones, as they do on the web.

We are interested in formal accounts of obligations that build on Searle's notion of construction of social reality. The obvious candidate for a formalization of norms and obligations is deontic logic, the logic of obligations. In particular, we may use Anderson's reduction of $O(p)$, read as '$p$ is obligatory', to $\Box(\neg p \rightarrow V)$, read as either 'the absence of $p$ leads to a sanction' or 'the absence of $p$ leads to a bad state' [1]. Anderson's reduction has proven useful in agent theory as part of Meyer's reduction of deontic action logic to dynamic logic [19], in which $F(\alpha)$, to be read as 'action $\alpha$ is forbidden, is reduced to $[\alpha]V$, after the execution of $\alpha$ $V$ holds. However, these studies do not distinguish violations from sanctions, and they do not show how Searle's notion of social construction may fit in.

In this paper we introduce and study a deontic logic, using ideas developed in agent theory to formalize the notion of social construction. We formalize and extend an idea recently proposed by Boella and Lesmo [2]. They attribute mental attitudes to the normative system - which they therefore call the normative *agent*. They relate the external motivation of the agent to the internal motivation of the normative agent. The wishes (or desires, or goals) of the normative agent are the commands (or obligations) of the agent.

The relevance of this paper for agent theory is that it can be applied to several norm-based agent architectures that have recently been developed [2, 10, 12, 16]. The formalization of sanction-based obligations shows which are the motivations for the agents to fulfil the obligations they are subject to. In this way it is not necessary to assume that agents are designed so to fulfil obligations from which they do not gain any advantage. In this paper we also consider a decision-theoretic account of norms and obligations as an application of our results. Moreover, the relevance of our study for deontic logic is an alternative criticism to the weakening rule, which has already been criticized in the philosophical literature for its role in the Ross and Forrester paradoxes, and a criticism, based on legal reasoning, to the generally accepted conjunction rule.

We are motivated in this study by our research on norms for multiagent systems. In other works we propose obligations defined in a qualitative decision theory inspired by the BOID architecture of Broersen *et al.* [10]. In this paper we study logical relations between such obligations. This paper is thus a kind of analysis of an element of the model we present in [3, 8], which extends Boella and Lesmo's definition of sanction-based obligations, and distinguish between what counts as a violation, and which sanctions are applied: the agents take a decision based on the recursive modelling of the behavior of the normative

agent according to whether it sanctions them or not. In [5, 4, 6] the same model is used to formalize policies regulating virtual communities on the web.

## 2 Social Constructions

We have to fix some terminology. First, we identify normative systems with normative agents and switch freely between them. This is based on the attribution of mental attitudes to the normative system, as discussed in the introduction [3, 8]. We specifically do not restrict the normative system or normative agent to human agents. Second, during the last two decades, knowledge and goals have been replaced by beliefs, desires and intentions. Since for the normative agent we have to choose between goals and desires we opt for the latter, though in the context of this paper desires and goals can be interchanged (see Section 2.3).

Our method to formalize obligations is modal logic [13]. Assume a modal language that contains at least the modalities $O_{AN}(p)$: in normative system $N$, agent A is obliged to see to it that $p$, $D_N(p)$: the normative agent desires $p$, $V_{NA}(p)$: according to $N$, $p$ counts as a violation by $A$, and $S_{NA}(p)$: according to $N$, $A$ is sanctioned for $p$. The following two choices determine our deontic logic. First, the definition of $O_{AN}$ in terms of $D_N$, $V_{NA}$ and $S_{NA}$. An agent is obliged to see to it that $p$ iff the normative agent desires that $p$, the normative agent desires that $\neg p$ counts as a violation by $A$, and the normative agent desires that if $\neg p$ counts as a violation, then $A$ is sanctioned for $\neg p$. Note that an obligation for $p$ implies that the normative agent has a desire for $p$, but this does not imply that all agents have an obligation for $p$. For the other agents absence of $p$ does not have to count as a violation. Moreover, the fact that $\neg p$ counts as a violation is not a fact independent from the normative agent's behavior: rather, it is a desire of N, so that it must decide to do something for making $\neg p$ count as violation.

Given this definition, in case of a violation, it is possible to predict N's behavior from his desires and goals: he will decide that $\neg p$ is a violation and he will sanction A. Second, the logical properties of $D_N$, $V_{NA}$ and $S_{NA}$. Instead of choosing one particular logic for these three primitive concepts, which would lead to a unique deontic logic for a particular definition, in this paper we take the logical properties of $D_N$, $V_{NA}$ and $S_{NA}$ as a parameter. That is, we show that $O_{AN}$ has a certain property if $D_N$, $V_{NA}$ and $S_{NA}$ have certain other properties. In this way our results can be applied to a wide variety of logical systems.

Boella and Lesmo's construction introduces a new problem, which may be called the obligation distribution problem. Given a set of goals or desires of the normative agent, how are they distributed as obligations over the agents? Typical subproblems which may be discussed here are whether a group of agents can be jointly obliged to see to something, without being individually obliged. Similar problems are studied, e.g., by [11]. Another subproblem is whether agents can transfer their obligations to other agents. In this paper we do not study these questions, and we simply define that a desire of the normative agent counts as an obligation of agent $A$, when the unfulfillment of this desire counts as a violation by agent $A$.

### 2.1 Counts as a Violation

In Searle's theory, *counts as* is a conditional relativized to an institution or society. Thus, when $p$ and $q$ are descriptions of some state of affairs or action, and $N$ is a description of an institution, then $p \Rightarrow_N q$ may be read as '$p$ counts as $q$ according to institution $N$'. A conditional logic along this line has been developed by Jones and Sergot [17]. Jones and Sergot study the *counts as* conditional $p \Rightarrow_i q$ in the context of modal action logic $E_a(p)$ for agent $a$ sees to it that $p$. The conditional $p \Rightarrow_i q$ is closed under left-OR, right-AND, and TRANS, but not under right-W nor left-S. The latter makes their conditional a defeasible one. Their motivation is that their action operator satisfies the success postulate $E_a(p) \rightarrow p$, and that they do not like to infer $E_y(E_x(A)) \Rightarrow_i B$ from $E_x(A) \Rightarrow_i B$.

In a normative system with norms $\{n_1, \ldots, n_k\}$, with $p$ and $N$ as before, $n$ a norm and $V$ as a violation operator, $p \Rightarrow_N V_A(n)$ may be read as '$p$ counts as a violation by agent A according to norm $n$ of institution $N$'. However, in deontic logic the formal language usually abstracts away from agents, institutions and explicit norms, because either they are irrelevant for the logical relations between obligations, or they seem to block such an analysis. In Section 2 and 3 we also abstract away from the explicit norms, such that '$p$ counts as a violation' may be represented as $p \Rightarrow_N V_{NA}$, which we abbreviate by $V_{NA}(p)$. In Section 4 we discuss explicit normative systems. For an extensive discussion for and against explicit norms in deontic logic see the discussion on the so-called diagnostic framework for deontic reasoning (diOde) in [24].

There is no consensus on the logical properties of the *counts as* conditional, maybe because the conditional can be used in many different kinds of applications. We therefore do not build our analysis on the conditional. The approach we follow in this paper is to study a default interpretation of $V_{NA}(p)$, together with various other alternatives. That is, a particular interpretation of it will be used by default, in absence of information to the contrary.

For our default interpretation, we say that the following property called strengthening (S) holds, whereas the property called weakening (W) does not hold. For example, if speeding counts as a violation, then speeding in a red car counts as a violation too. However, if driving under 18 counts as a violation, then driving by itself does not count as a violation. Note that the property called conjunction (AND) follows from S. We write $\rightarrow$ for the material implication.

S $\quad \vdash V_{NA}(p) \rightarrow V_{NA}(p \wedge q)$
not-W $\nvdash V_{NA}(p) \rightarrow V_{NA}(p \vee q)$
AND $\quad \vdash V_{NA}(p) \wedge V_{NA}(q) \rightarrow V_{NA}(p \wedge q)$

If both S and W hold, then we have $V_{NA}(p) \rightarrow V_{NA}(q)$, i.e., when some formula counts as a violation, then all formulas count as a violation. In other words, in such a case the logic only distinguishes between no violation and violation. In such a case, we say that the 'counts as a violation' operator $V_{NA}$ trivializes. This trivial operator $V_{NA}$ corresponds to the notion of violability studied by Anderson [1], because it does not distinguish between distinct violations (see e.g., [24]). Note that this kind of trivialization should be distinguished from the

trivialization represented by $p \leftrightarrow V_{NA}(p)$. In the latter kind of trivialization, the modal operator has become superfluous. In our kind of trivialization, we go from a fine-grained to binary distinction.

Moreover, by default we assume that the following property called disjunction (OR) holds. For example, assume that 'driving 120 km/hour' counts as a violation and that 'driving drunk' counts as a violation. By default we conclude that 'driving drunk or 120 km/hour' counts as a violation, because we know that some norm has been violated.

OR $\vdash V_{NA}(p) \wedge V_{NA}(q) \rightarrow V_{NA}(p \vee q)$

Clearly, for our default interpretation we cannot use standard normal modal operators, because they satisfy W. This suggests the use of a minimal modal logic, as used in several recent agent logics [18]. However, when $\Box$ is a normal modal operator, then $\Box'$ defined by $\Box'(p) =_{\text{def}} \Box(\neg p)$, satisfies S instead of W. This definition in terms of a normal modal operator is the default choice for $V_{NA}$. We say that $\Box'$ is the negation or negative of $\Box$. For example, prohibitions are the negative of obligations. Note that permission $P(p) =_{\text{def}} \neg O(\neg p)$ is also sometimes called the negation (or dual) of an obligation. Thus, what we call the negation should be distinguished from other uses in the literature.

### 2.2 Being Sanctioned

Sanctioning is an action of the normative agent. The normative agent sanctioning A for $\neg p$ with $s$ due to norm $n$ may be represented by $S_{NA}(\neg p, s, n)$. A logical property we discuss later in this paper is that the normative agent can sanction only if the agent's behavior counts as a violation of this norm.

Whether an action of the normative agent is a sanction or just any other action, i.e., whether it counts as a sanction, is also a social construction. For example, whether giving a fine counts as a sanction for late delivery, may depend on a convention in the society. We may thus write $s \Rightarrow_N S_{NA}(\neg p, n)$: according to institution $N$, $s$ counts as a sanction for $\neg p$, agent $A$ and norm $n$. However, it is important to notice that $S_{NA}(\neg p)$ in the definition of obligation should not be read as '$\neg p$ counts as a sanction'. The normative agent does not desire that $s$ counts as a sanction, but that $\neg p$ is sanctioned with $s$. This is subtly different.

If we abstract away from norm $n$ and sanction $s$, then we write $S_{NA}(\neg p)$ for $\neg p$ is being sanctioned. As far as we know, this operator has not been discussed in the literature. Again, it seems reasonable to accept, as a starting point, S, AND, and OR, and reject W. This is therefore our default choice.

S $\vdash S_{NA}(p) \rightarrow S_{NA}(p \wedge q)$
not-W $\nvdash S_{NA}(p) \rightarrow S_{NA}(p \vee q)$
AND $\vdash S_{NA}(p) \wedge S_{NA}(q) \rightarrow S_{NA}(p \wedge q)$
OR $\vdash S_{NA}(p) \wedge S_{NA}(q) \rightarrow S_{NA}(p \vee q)$

Alternatively, we may abstract away from the reason for the sanction, and write $S_{NA}(s)$ for $A$ is sanctioned with $s$. The latter can also be simplified to a single proposition $s$.

### 2.3 Desires

There has been some discussion on the distinction between desires and goals. If we consider a deliberation cycle, then desires are usually considered to be more primitive, because goals have to be *adopted* [14] or *generated* [10]. Goals can be based on desires, but also on other sources. For example, a social agent may adopt as a goal the desires of another agent, or an obligation. In knowledge based systems [20], goals are related to utility aspiration level and to limited (bounded) rationality. Moreover, here goals have desirability aspect as well as intentionality aspect, whereas in BDI circles it has been argued that this desirability aspect should be separated.

An important distinction for our present purposes is whether we may have $D_N(p)$ and $D_N(\neg p)$ at the same time. If such conflicts are considered to be inconsistent, then the desires can be formalized by a normal modal operator of type KD. System KD is the smallest set that contains the propositional formulas, the axioms $K : D_N(p \rightarrow q) \rightarrow (D_N(p) \rightarrow D_N(q))$ and $D : \neg(D_N(p) \wedge D_N(\neg p))$, and is closed under modus ponens and necessitation. This is the formalization used in e.g., [21], and our default choice.

If desires are allowed to conflict, and $D_N(p) \wedge D_N(\neg p)$ has to be represented in a consistent way, then desires may be represented by a so-called minimal modal operator [13, 18], in which the conjunction rule AND is not valid.

## 3 Obligations

### 3.1 Basic Definition

We start with the definition of obligations in terms of desires, counts as a violation, and being sanctioned. The basic definition contains three clauses. (1) says that an obligation of $A$ is a desire of $N$. (2) says that if $\neg p$ is the case, then $N$ desires that it counts as a violation. (3) says that if $\neg p$ counts as a violation, then $N$ desires that it is sanctioned. Permissions are defined as usual.

**Definition 1 (Obligation).** *Consider a modal logic with modal operators $D_N$ (for desire or goal), $V_{NA}$ (for counts as a violation) and $S_{NA}$ (for being sanctioned). Obligation and permission are defined by:*

$$
\begin{aligned}
O_{AN}(p) =_{def}\ & D_N(p) \wedge && (1)\\
& \neg p \rightarrow D_N(V_{NA}(\neg p)) \wedge && (2)\\
& V_{NA}(\neg p) \rightarrow D_N(S_{NA}(\neg p)) && (3)\\
P_{AN}(p) =_{def}\ & \neg O_{AN}(\neg p)
\end{aligned}
$$

We now consider various properties for the three modal operators of the normative agent. We first consider the case in which the three modal operators are defined as either modal operators of type KD or negatives of them.

**Proposition 1.** *Let the modal operator $D_N$ be a normal modal operator of type KD, and let $V_{NA}$ and $S_{NA}$ be negated operators of type KD in the sense*

*that $V_{NA}\neg$ and $S_{NA}\neg$ are normal modal operators of type KD. The logic does not satisfy weakening (W), strengthening (S), conjunction (AND), or disjunction (OR). It only satisfies the following formula called Deontic (D):*

| | |
|---|---|
| *not-S* | $\nvdash O_{AN}(p) \rightarrow O_{AN}(p \wedge q)$ |
| *not-W* | $\nvdash O_{AN}(p) \rightarrow O_{AN}(p \vee q)$ |
| *not-AND* | $\nvdash O_{AN}(p) \wedge O_{AN}(q) \rightarrow O_{AN}(p \wedge q)$ |
| *not-OR* | $\nvdash O_{AN}(p) \wedge O_{AN}(q) \rightarrow O_{AN}(p \vee q)$ |
| *D* | $\vdash O_{AN}(p) \rightarrow P_{AN}(p)$ |

*Proof.* *AND does not hold due to (2), and W and OR do not hold due to (3).*

The following proposition studies in more detail the conditions under which the properties are satisfied.

**Proposition 2.** *$O_{AN}$ does not satisfy S.*

*$O_{AN}$ satisfies W if $D_N$ satisfies W, $V_{NA}$ trivializes in the sense that it satisfies W as well as S, and $S_{NA}$ satisfies W.*

*$O_{AN}$ satisfies AND if $D_N$ satisfies AND and W, $V_{NA}$ trivializes, and $S_{NA}$ satisfies OR.*

*$O_{AN}$ satisfies OR if $D_N$ satisfies OR and AND, $V_{NA}$ satisfies W and AND, and $S_{NA}$ satisfies AND.*

*$O_{AN}$ satisfies D if $D_N$ satisfies D.*

**Corollary 1.** *$O_{AN}$ satisfies W, AND and OR if $D_N$ is a normal modal operator, $V_{NA}$ trivializes and $S_{NA}$ is the negative of a normal modal operator.*

### 3.2 Interpretation of Results

The corollary explains why Anderson's reduction, as well as most deontic logics developed along this line, only consider a single violation constant. Such a simple notion of violability leads to a logic with many desirable properties. Let us consider the results in more detail.

In so-called Standard Deontic Logic (SDL), a normal modal system of type KD, the obligations satisfy weakening and conjunction, but lack strengthening. The result that our $O_{AN}$ lacks weakening is thus in conflict with this logic, but it is in line with a long standing tradition in deontic logic that rejects it, see [23] for a survey and discussion. The reason is that this proof rule leads to counterintuitive results in the so-called Ross paradox ('you ought to mail the letter' implies that 'you ought to mail the letter or burn it') and the Forrester paradox ('you should not kill', but 'if you do so then you should do it gently'). However, here the reason is completely different. W does not hold due to (3), which means that the reason is not the violability but the association of sanctions with violations.

The result that $O_{AN}$ lacks conjunction is surprising, because most deontic logics satisfy this rule. The motivation of deontic logics not satisfying this rule is that they want to represent conflicts in a consistent way. Moreover, our result is in particular surprising since the rule is already blocked by clause (2), i.e.,

it is blocked due to the violability clause. The reason is the condition of (2). For an example, consider the two obligations 'driving 120 km/hour' counts as a violation and 'driving drunk' counts as a violation. In the logic, if we have that 'either someone drives 120 km/hour or he drives drunk', then this does *not* count as a violation. This phenomena can also be observed in reality. For example, in many legal courts someone cannot be sentenced if it is not clear which norm has been violated. There is only a violation if the norm which is violated can be identified. In such circumstances, if someone has committed a violation, but we do not know which one, then we cannot sanction him.

### 3.3 Two Variants that Disturb AND

There are several issues in this formalization of obligation in Definition 1. For example, the three conditions informally given in the introduction can be represented in another way, and additional conditions can be added. However, from the perspective of our logical analysis, all changes we have considered only lead to minor variations of the two propositions, and they do not interfere with the analysis. The following two definitions imply a small change to Proposition 2.

First, the formalization of 'the absence of $p$'. In clause (2), the absence of $a$ is represented by $\neg a$. Consequently, if nothing is known then it does not count as a violation. An alternative way to formalize it is to use $not(a)$, where *not* is the negation by failure as used in logic programming.

Second, introduction of a particular perspective, such as the perspective of an external observer, of agent A or of the normative agent. For example, if everything is considered from the perspective of agent A, then we may write:

**Definition 2 (Subjective Obligations).** *Consider a modal logic as before, with additionally a normal modal operator $B_A$ for 'agent A believes ...'. Agent A believes to be obliged to see to it that p iff:*

$$
\begin{aligned}
BO_{AN}(p) =_{def} \quad & B_A(D_N(p)) \wedge && (1)\\
& B_A(\neg p) \rightarrow B_A(D_N(V_{NA}(\neg p))) \wedge && (2)\\
& B_A(V_{NA}(\neg p)) \rightarrow B_A(D_N(S_{NA}(\neg p))) && (3)
\end{aligned}
$$

Clearly, for obligations based on *not* operator and for subjective obligations, Proposition 1 still holds. Proposition 2 also holds, with the minor adaptation that AND no longer holds under these conditions (nor under any other reasonable conditions).

Moreover, for various variations of Definition 2, for example the one in which (2) would read $B_A(\neg p \rightarrow D_N(V_{NA}(\neg p)))$, Proposition 2 still valid, but for other variations, such as the one in which (2) reads $B_A(B_N(\neg p) \rightarrow D_N(V_{NA}(\neg p)))$, the adapted proposition holds. Summarizing, our analysis can directly be applied to such subjective obligations.

### 3.4 Four Equivalent Variations

In this section we discuss four more variations to the central definition, which do not influence our result.

First, the formalization of 'if ... then ...' structures. In clause (2) and (3), they are represented by a material implication (within the desire modality), whereas it is well known that this is problematic. However, other conditional logics proposed in the literature are weaker than the material implication, such that the logic of $O_{AN}$ can only become weaker.

Second, additional clauses that represent realism and other borderline cases. For example, we may add a clause that $O_{AN}(p)$ implies that $p$ is consistent, or that $O_{AN}(p)$ implies that $\neg p$ is consistent. Such borderline cases do not influence the two propositions in any significant way.

Third, additional clauses that distinguish goals from desires (i.e., by introducing besides desires also goals), require that the normative agent does not desire violations (or desires that there are no violations), assume that the normative agent has at least one way to apply the sanction, etc. Again, for any reasonable additional clauses we have considered, such additional clauses only make the logic of $O_{AN}$ weaker.

Fourth, in the following definition sanctions are made explicit. That is, we may say not only that $\neg p$ is sanctioned but also which sanction is applied. This leads to the introduction of an additional clause which says that the normative agent does not desire to apply the sanction anyway, i.e., even without a violation. Such rare cases are known, of course, but they are excluded in our model. The formalization of this new clause seems not completely satisfactory. We would have like to add the unconditional $D_N(\neg s)$. However, this unconditional clause is incompatible with our interpretation of $D_N$ as a normality, because (3) and (4) together would imply $\neg V_{NA}(\neg p)$. In other words, (4) can only be formalized by $D_N(\neg s)$ if we adopt for $D_N$ a non-normal modality, or a non-monotonic logic.

**Definition 3 (Modal Logic with Explicit Sanctions).** *Consider a modal logic with modal operators $D_N$ (for desire or goal) and $V_{NA}$ (for counts as a violation). Obligation with explicit sanction is defined by:*

$$
\begin{aligned}
O_{AN}(p,s) =_{def}\ & D_N(p)\wedge && (1)\\
& \neg p \rightarrow D_N(V_{NA}(\neg p))\wedge && (2)\\
& V_{NA}(\neg p) \rightarrow D_N(s)\wedge && (3)\\
& \neg V_{NA}(\neg p) \rightarrow D_N(\neg s) && (4)
\end{aligned}
$$

For fixed $s$, Proposition 1 and 2 both still hold, when in the latter the conditions on $S_{NA}$ are dropped.

## 4 Decision Theory

### 4.1 Normative Systems

This section illustrates an area where our theory can be applied. The logical analysis has shown that there are many ways to formalize obligations in a modal logic of desires, counts as and being sanctioned. However, in the logical analysis of such obligations, the following pattern emerges. If $V_{NA}$ does not trivialize, then the logic does not satisfy several proof rules which are often accepted in deontic logic. Now consider the following definition of a normative system.

**Definition 4.** *Let $L$ be a propositional language. A normative system is a tuple $\langle N, V, S\rangle$ in which $N = \{n_1, \ldots, n_k\}$ is a set of norms, $V$ is a function that associates with every norm a formula of $L$ called its violation, and $S$ is a function that associates with every norm a propositional formula called its sanction.*

In this setting, we may say that the normative system implies the obligation $O(p, s)$ if there is a norm whose violation condition is $\neg p$. However, it is not very clear what the logical relations between these norms are, and what other methods we have to analyze the properties of such a system. If the norms are closed under for example weakening, then if this system would contain a norm with violation condition $p$ would be equivalent to a normative system which contains the same norm, and moreover a norm with violation condition $p \wedge q$. Moreover, if the system is closed under conjunction and the system contains a norm with violation condition $r$, then the system is equivalent to a normative system which in addition contains a norm with violation condition $p \vee r$. But what does this equivalence mean? Moreover, such an account does not take the sanctions into account.

We propose the following idea. Given a set of obligations. If for every decision making context, adding a new obligation to this set of obligations does not influence the decision making of the agent, then this new obligation is already implied (or *accepted*) by the set of obligations.

### 4.2 Decisions

In this section we introduce decisions in the logic.

**Definition 5 (Decision).** *Let the atomic variables be partitioned into three sets $A$, the decision variables of agent $A$, $N$, the decision variables of the normative agent, and $P$, the parameters. A state of the world $w$ is a propositional sentence. A decision $d$ of agent $A$ ($N$) in state $w$ is a propositional formula built from $A$ ($N$) only, such that $w \wedge d$ is consistent.*

We make several strong assumptions. A full qualitative decision theory has to incorporate a way to encode consequences of decisions. If we assume complete knowledge, i.e., the state of the world implies a truth value for each parameter, then we do not have to consider such effects, because effectively we only reason with ought-to-do obligations. An obligation $O_{AN}(p)$ is an ought-to-do obligation if $p$ contains variables of $A$ only, and an ought-to-be obligation otherwise.

**Definition 6.** *A state of the world contains complete knowledge if it implies either each variable of $P$, or its negation.*

With this new machinery, we can formalize the condition that the normative agent has a way to apply the sanction. We may formalize a new variant of our definition of obligation, with an additional clause is thus that there is a decision of N such that this decision implies sanction $s$. In our case, this means that $s$ is a decision variable of N.

### 4.3 Decision Rule

To evaluate its decisions, an agent may either consider the violations or the sanctions. This represents different agent types: an obedient or respectful agent considers its violations, whereas a selfish agent may only consider the sanctions.

**Definition 7 (Decision Evaluation).** *Let $w$ be a state of the world and $d$ a (partial) decision. The set of violated norms is $Viol(w, d) = \{n \in N \mid w \wedge d \vdash V(n)\}$ and the set of sanctions is $Sanc(w, d) = \{S(n) \mid n \in Viol(w, d)\}$.*

The evaluations are used in the agent's decision rule, assuming that all sanctions have the same cost:

**Definition 8 (Decision Rule).** *Given state of the world $w$. An obedient agent selects a decision $d$ that minimizes (with respect to set inclusion) $Viol(w, d)$. A selfish agent minimizes (with respect to set inclusion) the logical closure of $Sanc(w, d)$.*

### 4.4 Acceptance

We analyze the normative system using the notion of acceptance.

**Definition 9.** *Given an agent type. A normative system accepts an obligation $O(p, s)$ if for any state of the world, adding to the normative system the norm $n$ with violation $V(n) = \neg p$ and sanction $S(n) = s$, does not change the optimal decisions.*

We can consider the logical properties of the acceptance condition by abstracting away from the normative systems. The following proposition implies that the set of accepted obligations is not closed under weakening, strengthening, or conjunction. The results are in line with our logical analysis.

**Proposition 3.** *There is a normative system that accepts $O(a)$ but not $O(a \vee b)$ or $O(a \wedge b)$, and there is a normative system that accepts $O(a)$ and $O(b)$ but not $O(a \wedge b)$ or $O(a \vee b)$.*

## 5 Summary

In this paper we obtain the following results.

- We propose a logical framework to study social constructions.
- We define obligations in terms of this social construction, and study its properties.
- We define acceptance relations for normative systems.
- We contribute to the philosophical discussions on the distinction between violations and sanctions in Anderson's reduction, and between implicit and explicit normative systems.

Further relations between deontic logic and the theory of normative systems is subject of ongoing research, e.g., in [7] we consider the notion of strong permission. In [8] we consider the problem of norm creation.

## References


[1] A. Anderson. A reduction of deontic logic to alethic modal logic. *Mind*, 67:100–103, 1958. 28, 30

[2] G. Boella and L. Lesmo. A game theoretic approach to norms. *Cognitive Science Quarterly*, 2(3-4):492–512, 2002. 28

[3] G. Boella and L. van der Torre. Attributing mental attitudes to normative systems. In *Procs. of AAMAS'03*, Melbourne, 2003. ACM Press. 28, 29

[4] G. Boella and L. van der Torre. Decentralized control obligations and permissions in virtual communities of agents. In *Procs. of ISMIS'03*, 2003. Springer Verlag. 29

[5] G. Boella and L. van der Torre. Local policies for the control of virtual communities. In *Procs. of IEEE/WIC WI'03*, 2003. 29

[6] G. Boella and L. van der Torre. Norm governed multiagent systems: The delegation of control to autonomous agents. In *Procs. of IEEE/WIC IAT'03*, 2003. 29

[7] G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *Procs. of ICAIL'03*, Edinburgh, 2003. ACM Press. 37

[8] G. Boella and L. van der Torre. Rational norm creation: Attributing Mental Attitudes to Normative Systems, Part 2. In *Procs. of ICAIL'03*, Edinburgh, 2003. ACM Press. 28, 29, 37

[9] M.Bratman. *Intentions, plans, and practical reason*. Harvard University Press, Harvard (MA), 1987. 27

[10] J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002. 28, 32

[11] J. Carmo and O. Pacheco. Deontic and action logics for collective agency and roles. In *Proc. Fifth International Workshop on Deontic Logic in Computer Science (DEON'00)*, pages 93–124, 2000. 29

[12] C.Castelfranchi, F. Dignum, C. M. Jonker, and J. Treur. Deliberate normative agents: Principles and architecture. In *Intelligent Agents VI - Procs. of ATAL'99*, 2000. Springer Verlag. 28

[13] B. Chellas. *Modal logic: an introduction*. Cambridge University Press, Cambridge (UK), 1980. 29, 32

[14] R. Conte, C. Castelfranchi, and F. Dignum. Autonomous norm-acceptance. In *Intelligent Agents V - Procs of ATAL'98*, pages 319–333. 1999. Springer Verlag. 32

[15] D. Dennett. *The intentional stance*. Bradford Books, Cambridge (MA), 1987. 27

[16] F. Dignum, D. Morley, E. A. Sonenberg, and L. Cavedon. Towards socially sophist icated BDI agents. In *Procs. of ICMAS'00*, pages 111–118, Boston, 2000. 28

[17] A. Jones and M. Sergot. A formal characterisation of institutionalised power. *Journal of IGPL*, 3:427–443, 1996. 30

[18] S. Kraus, K. Sycara, and A. Evenchik. Reaching agreements through argumentation; a logical model and implementation. *Artificial Intelligence*, 104:1–69, 1998. 31, 32

[19] J. J. Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame J. of Formal Logic*, 29(1):109–136, 1988. 28

[20] A. Newell and H. Simon. *Human Problem Solving*. Prentice-Hall, 1972. 27, 32

[21] A. Rao and M. Georgeff. Modeling rational agents within a BDI architecture. In *Procs. of KR'91*, pages 473–484. 1991. Morgan Kaufmann. 32

[22] J. Searle. *The Construction of Social Reality*. The Free Press, New York, 1995. 28

[23] L. van der Torre and Y. Tan. Contrary-to-duty reasoning with preference-based dyadic obligations. *Annals of Mathematics and AI*, 27:49–78, 1999. 33

[24] L. van der Torre and Y. Tan. Diagnosis and decision making in normative reasoning. *Artificial Intelligence and Law*, 7(1):51–67, 1999. 30

# Automatically Decomposing Configuration Problems*

Luca Anselma, Diego Magro, and Pietro Torasso

Dipartimento di Informatica
Università di Torino
Corso Svizzera 185; 10149 Torino; Italy
{anselma,magro,torasso}@di.unito.it

**Abstract.** Configuration was one of the first tasks successfully approached via AI techniques. However, solving configuration problems can be computationally expensive. In this work, we show that the decomposition of a configuration problem into a set of simpler and independent subproblems can decrease the computational cost of solving it. In particular, we describe a novel decomposition technique exploiting the compositional structure of complex objects and we show experimentally that such a decomposition can improve the efficiency of configurators.

## 1 Introduction

Each time we are given a set of components and we need to put (a subset of) them together in order to build an artifact meeting a set of requirements, we actually have to solve a *configuration problem*. Configuration problems can concern different domains. For instance, we might want to configure a PC, given different kinds of CPUs, memory modules, and so on; or a car, given different kinds of engines, gears, etc. Or we might also want to configure abstract entities in non-technical domains, such as students' curricula, given a set of courses.

In early eighties, configuration was one of the first tasks successfully approached via AI techniques, in particular because of the success of $R1/XCON$ [10]. Since then, various approaches have been proposed for automatically solving configuration problems. In the last decade, instead of heuristic methods, research efforts were devoted to single out formalisms able to capture the system models and to develop reasoning mechanisms for configuration. In particular, configuration paradigms based on Constraint Satisfaction Problems (CSP) and its extensions [12, 13, 1, 18] or on logics [11, 3, 16] have emerged.

In the rich representation formalisms able to capture the complex constraints needed in modeling technical domains, the configuration problem is theoretically intractable (at least NP-hard, in the worst case) [5, 15, 16]. Despite the theoretical complexity, many real configuration problems are rather easy to solve [17]. However, in some cases the intractability does appear also in practice and solving some configuration problems can require a huge amount of CPU time. These

* This work has been partially supported by ASI (Italian Space Agency).

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 39–52, 2003.
© Springer-Verlag Berlin Heidelberg 2003

ones are rather problematic situations in those tasks in which low response time is required. E.g. in interactive configuration the response time should not exceed a few seconds and on-line configuration on the Web imposes even stricter requirements on this configurator feature.

There are several ways that can be explored to control computational complexity in practice: among them, making use of off-line knowledge compilation techniques [14]; providing the configurator with a set of domain-specific heuristics, with general focusing mechanisms [6] or with the capability of re-using past solutions [4]; defining techniques for automatically decomposing a problem into a set of simpler subproblems [9, 8]. These approaches are not in alternative and configurators can make use of different combinations of them. However it makes sense to investigate to what extent each one of them can contribute to the improvement of the efficiency of configurators. In the present work, we focus on automatic problem decomposition, since to the best of our knowledge this issue has not received much attention in the configuration community.

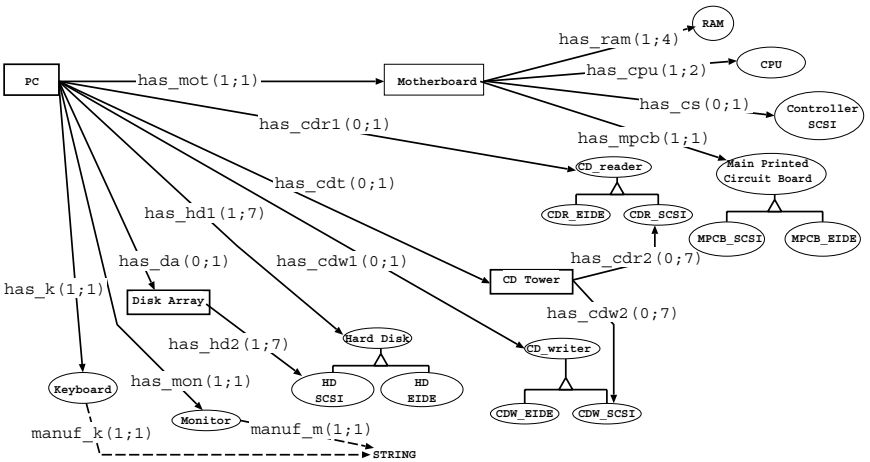
In [7] a structured logical approach to configuration is presented. Here we commit to the same framework as that described there and we present a novel problem decomposition mechanism that exploits the knowledge on the compositional structure (i.e. the knowledge relevant to parts and subparts) of the complex entities that are configured. We also report some experimental results showing its effectiveness.

Section 2 contains an overview of the conceptual language, while Section 3 defines configuration problems and their solutions. In Section 4 a formal definition of the *bound* relation, which problem decomposition is based on, is given; moreover, in that same section, a configuration algorithm making use of decomposition is reported and illustrated by means of an example. Section 5 reports the experimental results, while Section 6 contains some conclusions and a brief discussion.

## 2 Conceptual Language

In the present paper the $\mathcal{FPC}$ (Frames, Parts and Constraints) [7] language is adopted to model the configuration domains. Basically, $\mathcal{FPC}$ is a frame-based KL-One like formalism augmented with a constraint language.

In $\mathcal{FPC}$, there is a basic distinction between *atomic* and *complex* components. *Atomic components* are the basic building blocks of configurations and they are described by means of properties, while *complex components* are structured entities whose characterization is given in terms of subparts which can be complex components in their turn or atomic ones. $\mathcal{FPC}$ offers the possibility of organizing classes of (both atomic and complex) components in *taxonomies* as well as the facility of building *partonomies* that (recursively) express the *whole-part relations* between each complex component and its (sub)components. A set of *constraints* restricts the set of valid combinations of components and subcomponents in configurations. These constraints can be either specific to the modeled domain or derived from the user's requirements.




**CONSTRAINTS**

**Associated with *PC* class**

```
"In any PC, if there is a EIDE main printed circuit board and at least one SCSI device,
 then there must be a controller SCSI"
[co1](<has_mot,has_mpcb>)(in MPCB_EIDE) AND
      ((<has_hd1>)(in HD_SCSI(1;7)) OR (<has_cdr1>)(in CDR_SCSI(1;1)) OR
       (<has_cdw1>)(in CDW_SCSI(1;1))
     )==>(<has_mot,has_cs>)(1;1)
```

**Associated with *Motherboard* class**

```
"In any motherboard, if there is a SCSI main printed circuit board,
 then there should be no controller SCSI"
[co2](<has_mpcb>)(in MPCB_SCSI)==>(<has_cs>)(0;0)
```

**Associated with *CD Tower* class**

```
"In any CD tower, there must be at least one CD reader or CD writer"
[co3](<has_cdr2>,<has_cdw2>)(1;14)
```

**Fig. 1.** A simplified PC conceptual model ($CM_{PC}$)

We illustrate $\mathcal{FPC}$ by means of an example; for a formal description, refer to [7]. In fig. 1 a portion of a simplified conceptual model relevant to PC configuration is represented. The classes of complex components (e.g. *PC*, *Motherboard*, ...) are represented as rectangles, while classes of atomic components (e.g. *Main Printed Circuit Board*, *CD_reader*, ...) are represented as ellipses. *Partonomic roles* represent whole-part relations and are drawn as solid arrows. For instance, the *PC* class has the partonomic role *has_mot*, with minimum and maximum cardinalities 1, meaning that each PC has exactly one motherboard; partonomic role *has_cdr*1, whose minimum and maximum cardinalities are 0 and 1, respectively, expresses the fact that each PC can optionally have one CD reader, and so on. It is worth noting that the motherboard is a complex component having 1 to 4 RAM modules (see the *has_ram* partonomic role), one main printed circuit board (*has_mpcb* role), that can be either the SCSI or the EIDE type, etc.

*Descriptive roles* represent properties of components and they are drawn as dashed arrows. For example, the $Monitor$ component has a *string* descriptive role $manuf_m$, representing the manufacturer.

Each *constraint* is associated with a class of complex components and is composed by $\mathcal{FPC}$ predicates combined by means of the boolean connectives $\wedge, \vee, \neg, \rightarrow$. A *predicate* can refer to cardinalities, types or property values of (sub)components. The reference to (sub)components is either direct through partonomic roles or indirect through *chains of partonomic roles*. For example, in fig. 1 [$co2$] is associated with the $Motherboard$ class and states that, if $has_mpcb$ role takes values in $MPCB_SCSI$ (i.e. the main printed circuit board is the SCSI type), then $has_cs$ relation must have cardinality 0 (i.e. there must be no SCSI controller). An example of a chain of partonomic roles can be found in [$co1$]: the consequent of the constraint [$co1$] (associated with $PC$ class) states that the role chain $\langle has_mot, has_cs \rangle$ has cardinality 1, i.e. the $PC$ component has one Motherboard with one SCSI Controller. [$co3$] shows an example of a *union of role chains*: a component of type $CD\ Tower$ must have 1 to 14 $CD_reader$s *or* $CD_writer$s.

## 3 Configuration Problems

A *configuration problem* is a tuple $CP = \langle CM, T, c, C, V \rangle$, where $CM$ is a conceptual model, $T$ is a partial description of the complex object to be configured (the *target object*), $c$ is a complex component occurring in $T$ (either the target object itself or one of its complex (sub)components) whose type is $C$ (which is a class of complex objects in $CM$) and $V$ is a set of constraints involving component $c$. In particular, $V$ can contain the user's requirements that component $c$ must fulfill.

Given a configuration problem $CP$, the task of the configurator is to refine the description $T$ by providing a complete description of the component $c$ satisfying both the conceptual description of $C$ in $CM$ and the constraints $V$, or to detect that the problem does not admit any solution.

**Configuration Process** We assume that the configurator is given a main configuration problem $CP_0 = \langle CM, (c), c, C, REQS \rangle$, where $c$ represents the target object, whose initial partial description $T \equiv (c)$ contains only the component $c$; $REQS$ is the set of requirements for $c$ (expressed in the same language as the constraints in $CM$[1]). Therefore, the goal of the configurator is to provide a complete description of the target object (i.e. of an individual of the class $C$)

[1] It is worth pointing out that the user actually specifies her requirements in a higher level language (through a graphic interface) and the system performs an automatic translation into the representation language. This translation process may also perform some inferences, e.g. if the user requires a PC with a CD tower containing at least one CD reader and at least one CD writer, the system infers also an upper bound for the number of components of these two kinds, as in requirements $req3$ and $req4$ in fig. 2, where the upper bound 7 is inferred for both the number of CD readers and of CD writers that the CD tower can contain.

```
The manufacturer of the monitor must be the same as that of the keyboard
[req1](<has_mon,manuf_m>)=(<has_k,manuf_k>)

It must have a disk array
[req2](<has_da>)(1;1)

It must have a CD tower with at least one CD reader and at least one CD writer
[req3](<has_cdt,has_cdr2>)(1;7)
[req4](<has_cdt,has_cdw2>)(1;7)

It must have no more than 4 SCSI devices
[req5](<has_cdr1>,<has_cdw1>,<has_hd1>,<has_cdt,has_cdr2>,<has_cdt,has_cdw2>,
       <has_da,has_hd2>)(in CDR_SCSI ∪ CDW_SCSI ∪ HD_SCSI(0;4))
```

**Fig. 2.** User's Requirements for a PC ($REQS_{PC}$)

satisfying the model $CM$ and fulfilling the requirements $REQS$ (such a description is a *solution* of the configuration problem) or to detect that the problem does not admit any solution (i.e. that such an individual does not exist). Since $CM$ is assumed to be consistent, this last case happens only when the requirements $REQS$ are inconsistent w.r.t. $CM$. A sample description of an individual $PC$ satisfying the conceptual model $CM_{PC}$ in fig. 1 and fulfilling the requirements listed in fig. 2 is reported in fig. 4.f.

The configuration is accomplished by means of a search process that progressively refines the description of $c$. At each step the configuration process selects a complex component in $T$ (starting from the target object), it refines the description $T$ by inserting a set of direct components of the selected component (by choosing both the number of these components and their type) and then it configures all the direct complex components possibly introduced in the previous step. If, after a choice, any constraint (either in $CM$ or in $REQS$) is violated, then the process backtracks. The process stops as soon as a solution has been found or when the backtracking mechanism cannot find any open choice. In the last case, $CP$ does not admit any solution.

## 4 Decomposing Configuration Problems

Because of the inter-role constraints, both those in $CM$ and those in $REQS$, a choice made by the configurator for a component can influence the valid choices for other components. In [9, 8] it is shown that the compositional knowledge (i.e. the way the complex product is made of simpler (sub)components) can be exploited to partition the constraints that hold for a given component into sets in such a way that the components involved in constraints of two different sets can be configured independently. While such a decomposition has been proved useful in reducing the actual computational effort in many configuration problems, here we present an enhancement of such a decomposition mechanism that considers constraints as dynamic entities instead of static ones.

### 4.1 Bound and Unbound Constraints

The decomposition capability is based on a *bound* relation among constraints. We assume that, in any configuration, each individual component cannot be a direct part of two different (complex) components, neither a direct part of a same component through two different whole-part relations (*exclusiveness assumption on parts*).

Let $CP = \langle CM, T, c, C, V \rangle$ and $CONSTRS(C)$ be a configuration problem and the set of constraints associated with $C$ in $CM$, respectively and let $u, v, w \in V \cup CONSTRS(C)$. The *bound* relation $\mathcal{B}_c$ is defined as follows: **if** $P_u$ and $P_v$ are two predicates occurring in $u$ and in $v$, respectively, that mention both a *same* partonomic role $p$ of $C$ **then** $u\mathcal{B}_c v$ (i.e. if $u$ and $v$ refer, through their predicates, to a same part of $c$, then they are ***directly*** bound in $c$); **if** $u\mathcal{B}_c v$ **and** $v\mathcal{B}_c w$ **then** $u\mathcal{B}_c w$ (i.e. $u$ and $w$ are bound by ***transitivity*** in $c$). It is easy to see that $\mathcal{B}_c$ is an equivalence relation.

To solve $CP = \langle CM, T, c, C, V \rangle$, the configurator must refine the description of $c$ by specifying the set $COMPS(c)$ of its components and subcomponents. In particular, it specifies the type of each element in $COMPS(c)$ and, for each partonomic role occurring in the conceptual description of type $C$ (the type of component $c$) in $CM$, it specifies which elements in $COMPS(c)$ play that partonomic role.

If $S_1$ and $S_2$ are two different equivalence classes of constraints induced by the relation $\mathcal{B}_c$, let $COMPS_{S_1}(c)$ and $COMPS_{S_2}(c)$ be the sets of components in $COMPS(c)$ referred to by constraints in $S_1$ and in $S_2$, respectively. Given the exclusiveness assumption on parts, these two sets are disjoint and, for every pair of components $c_1 \in COMPS_{S_1}(c)$ and $c_2 \in COMPS_{S_2}(c)$, there is no constraint in $V \cup CONSTRS(C)$ linking them together. It follows that the choices of the configurator relevant to the components in $COMPS_{S_1}(c)$ do not interact with those relevant to the components in $COMPS_{S_2}(c)$. In other words, $S_1$ and $S_2$ represent two mutually independent configuration subproblems.

### 4.2 Decomposition Mechanisms

In fig. 3 a configuration algorithm making use of decomposition is sketched. For lack of space, let us illustrate the algorithm just by means of an example. Let's suppose that the user wants to configure a $PC$ (described by the conceptual model $CM_{PC}$ in fig. 1) meeting the set $REQS_{PC}$ of requirements stated in fig. 2.

At the beginning, the configurator is given the problem $CP_0 = \langle CM_{PC}, (pc1), pc1, PC, REQS_{PC} \rangle$. Besides the requirements $REQS_{PC}$, the set of constraints associated with $PC$ in $CM_{PC}$ are also considered to fully specify the problem (statement in row 3 of the algorithm in fig. 3). This initial situation is represented in fig. 4.a.

**Initial Decomposition Step** (statements in rows 5 and 6). Before starting the actual configuration process, the configurator attempts to decompose the

```
(1) configure(CM,T,c,C,V){
(2)   SUBPROBLEMS = <>;
(3)   - add to V the constraints associated with C in CM;
(4)   currentSP=V;
(5)   S=decompose(CM,T,c,currentSP);
(6)   for each s in S push(s, SUBPROBLEMS);
(7)   while(SUBPROBLEMS ≠ <>){
(8)     currentSP=pop(SUBPROBLEMS);
(9)     if(no choice made for the direct components of c involved in currentSP){
(10)       T = insertDirectComponents(CM,T,c,currentSP);
(11)       if(T== FAILURE) return FAILURE;
(12)    }else{
(13)       - choose a direct complex component d of c that has not been configured
             yet and that is involved in currentSP (let D be the type of d);
(14)       T=configure(CM,T,d,D,currentSP);
(15)       if(T==FAILURE) BACKTRACK;
(16)      }
(17)     - remove satisfied constraints from currentSP;
(18)     if(not solved currentSP){
(19)        currentSP=reviseConstraints(CM,c,currentSP);
(20)        S=decompose(CM,T,c,currentSP);
(21)        for each s in S push(s,SUBPROBLEMS);}
(22)   }//while
(23)   - complete T by inserting all the components and subcomponents of c not
         involved in the constraints in V
(24)   return T;
(25) }//configure
```

**Fig. 3.** Configuration algorithm overview

constraints that hold for the target object $pc1$. To do this, it partitions the constraints
$currentSP = [req1, \ldots, req5, co1]$ into a set of equivalence classes by computing the *bound* relation $\mathcal{B}_{pc1}$ in this set: it is easy to see that the constraints $req2, \ldots, req5, co1$ are bound in $pc1$ according to the definition of the *bound* relation. Instead, $req1$ is not bound with any other constraint belonging to $currentSP$. It follows that $currentSP$ can be partitioned into the two equivalence classes of contraints $S_1 = [req2, \ldots, req5, co1]$ and $S_2 = [req1]$, each one entailing a configuration subproblem.

**Resolution of Subproblems** (*while* statement in rows 7 to 22). These subproblems are mutually independent. One subproblem is chosen as the current one (in this example that one relevant to the constraints $S_1 = [req2, \ldots, req5, co1]$) and the other ones (in this example only that one relevant to $S_2 = [req1]$) are pushed into the $SUBPROBLEMS$ stack (see fig. 4.b).

**Insertion of Direct Components** (statement in row 10). To solve $S_1$ the configurator refines the description of the target object by inserting in it only those direct components of $pc1$ involved in the constraints relevant to the current subproblem. More precisely, the configurator considers each partonomic role $p$ of $PC$ class occurring in the constraints belonging to $S_1$ and makes for $p$ *two basic choices*: it chooses the number of direct components, playing the partonomic role $p$, to insert into the configuration and, for each one of them, it chooses its type. In this example, let's suppose that a CD reader, a CD writer, a hard disk (all of the SCSI type), a motherboard, a CD tower and a disk array are inserted

```
T1=(pc1)

SUBPROBLEMS = <>
currentSP = [req1,...,req5,co1]

a)
```

```
T1=(pc1)

SUBPROBLEMS = <S2 = [req1]>
currentSP = S1 = [req2,...,req5,co1]

b)
```

```
T2=(pc1 <has_cdr1 (cdr_scsi1)>
        <has_cdw1 (cdw_scsi1)>
        <has_hd1 (hd_scsi1)>
        <has_mot (mb1)>
        <has_cdt (cdt1)>
        <has_da (da1)>)

SUBPROBLEMS =
    <S2 = [req1],S12 = [req3,req4,req5]>
currentSP = S11 =[co1']

c)
```

```
T3=(pc1 <has_cdr1 (cdr_scsi1)>
        <has_cdw1 (cdw_scsi1)>
        <has_hd1 (hd_scsi1)>
        <has_mot (mb1 <has_mpcb (mpcb_scsi1)>
                      <has_cs ()>
                      <has_cpu (cpu1)>
                      <has_ram (ram1,ram2,ram3,ram4)>)>
        <has_cdt (cdt1)>
        <has_da (da1)>)

SUBPROBLEMS = <S2 = [req1]>
currentSP = S12 = [req3,req4,req5]

d)
```

```
T4=(pc1 <has_cdr1 (cdr_eide1)>
        <has_cdw1 (cdw_eide1)>
        <has_hd1 (hd_scsi1)>
        <has_mot (mb1 <has_mpcb (mpcb_scsi1)>
                      <has_cs ()>
                      <has_cpu (cpu1)>
                      <has_ram (ram1,ram2,ram3,ram4)>)>
        <has_cdt (cdt1 <has_cdr2 (cdr_scsi1)>
                       <has_cdw2 (cdw_scsi1)>)>
        <has_da (da1 <has_hd2 (hd_scsi2)>)>)

SUBPROBLEMS = <>
currentSP = S2 = [req1]

e)
```

```
T5=(pc1 <has_cdr1 (cdr_eide1)>
        <has_cdw1 (cdw_eide1)>
        <has_hd1 (hd_scsi1)>
        <has_mot (mb1 <has_mpcb (mpcb_scsi1)>
                      <has_cs ()>
                      <has_cpu (cpu1)>
                      <has_ram (ram1,ram2,ram3,ram4)>)>
        <has_cdt (cdt1 <has_cdr2 (cdr_scsi1)>
                       <has_cdw2 (cdw_scsi1)>)>
        <has_da (da1 <has_hd2 (hd_scsi2)>)>
        <has_mon (acme_mon1)>
        <has_k (acme_k1)>)

SUBPROBLEMS = <>
currentSP = S2 = []

f)
```

**Fig. 4.** A configuration example

into the current configuration (fig. 4.c). Since configuration is accomplished by means of a search process, it is worth pointing out that all the open choices (for instance, the alternative EIDE type for the CD reader, the CD writer and the hard disk, or the possibility of inserting more than one hard disk) have to be remembered as they may be explored as a consequence of a backtracking.

**Removal of Satisfied Constraints** (statement in row 17). The current tentative configuration $T_2$ does not contradict any constraint relevant to the current subproblem, moreover requirement $req2$ (imposing the existence of a disk array in the configured PC) is now satisfied and it can be removed from $currentSP$. The truth values of the other constraints belonging to $currentSP$ cannot be computed yet, since the configurator has not yet configured all the parts of the target object which these constraints refer to. For instance, a CD tower has been inserted into the current tentative configuration $T_2$, but it has not been configured yet; therefore, up to this point, it is impossible to know how many CD readers the CD tower will contain and thus the truth value of $req3$ is still unknown. Since $currentSP$ still contains some constraints (whose truth values

are unknown) referring to parts of some direct components of $pc1$ not yet considered by the configurator, the subproblem relevant to $currentSP$ is not solved yet.

**Further Decomposition Step** (rows 18 to 21). After having refined the description of $pc1$ with the insertion of some of its direct components, the configurator attempts a further decomposition of the current subproblem.

**Revision of Constraints and Re-computation of Bound Relation**. To perform this decomposition step, the configurator dynamically updates the form of the constraints in $currentSP$ (i.e. the constraints are treated as dynamic entities). In this sample case, even if the truth value of constraint $co1$ cannot be determined in the tentative configuration $T_2$, for some predicates occurring in $co1$ it is possible to say whether they are true or false. In particular, the predicates $(\langle has_hd1\rangle)(in\ HD_SCSI(1;7))$, $(\langle has_cdr1\rangle)(in\ CDR_SCSI(1;1))$ and $(\langle has_cdw1\rangle)(in\ CDW_SCSI(1;1))$ are all true in $T_2$. Therefore, in the context of the choices made by the configurator and that leaded to $T_2$, these predicates can be substituted by their truth values in $co1$ and $co1$ can be simplified in the following way:

$$[co1'](\langle has_mot, has_mpcb\rangle)(in\ MPCB_EIDE) \rightarrow (\langle has_mot, has_cs\rangle)(1;1).$$

Since the revision of the constraints relevant to the current subproblem may remove some predicates from the constraints (as it happens for $co1$ in this example), it may happen that some constraints that were previously bound have now become unbound, therefore it makes sense to compute the *bound* relation again, in this revised set of constraints. In our example, the relation $\mathcal{B}_{pc1}$ induces a partitioning of the revised set of constraints $currentSP = [req3, req4, req5, co1']$ into the two classes $S_{11} = [co1']$ and $S_{12} = [req3, req4, req5]$ of bound constraints. This means that *in the context of tentative configuration* $T_2$ (fig. 4.c), the current subproblem has been further decomposed into a set of independent subproblems.

**Resolution of Subproblems** (*while* statement in rows 7 to 22). As in the previous execution of the body of the *while*, the configurator chooses one subproblem as the current one (in this case, $currentSP = S_{11}$) while the other ones (in this case only that one relevant to $S_{12}$) have been pushed into the $SUBPROBLEMS$ stack. All the direct components of $pc1$ involved in the set $currentSP$ of constraints have already been inserted into the tentative configuration. To solve $S_{11}$, the motherboard $mb1$ needs to be configured: indeed, $co1'$ refers both to the main printed circuit board and to the optional SCSI controller, which are $mb1$ components (rows 13 to 15). This means solving the configuration problem $CP_{mb1} = \langle CM_{PC}, T2, mb1, Motherboard, \{co1'\}\rangle$.

The configuration of $mb1$ has to take into account both the set $S_{11}$ of constraints and constraint $co2$ associated with $Motherboard$ class in $CM_{PC}$ (fig. 1).

In this example, a SCSI main printed circuit board $mpcb_scsi1$ is inserted into the tentative configuration, therefore no SCSI controller is inserted (because of $co2$). To complete the configuration of $mb1$, the configurator inserts also a CPU ($cpu1$) and four memory modules (fig. 4.d). Constraint $co1'$ is now satisfied, thus it is removed from $currentSP$. Since $currentSP$ does not contain any

other constraint, the configuration of $mb1$ represents a solution to the current subproblem. The subproblem entailed by $S_{12} = [req3, req4, req5]$ becomes the current one.

This subproblem involves the $pc1$ direct complex components $cdt1$ and $da1$. It should be clear that there is no way of extending the tentative configuration $T3$ by configuring these two components while satisfying the constraints in $S_{12}$.

Indeed, $req3$ and $req4$ require that at least one CD reader and at least one CD writer are inserted into $cdt1$ and, given the conceptual model $CM_{PC}$, these two devices must be the SCSI type. The conceptual model states also that all the hard disks in the disk array are the SCSI type (and that there is at least one hard disk in a disk array). However, $T3$ already contains 3 SCSI devices; it follows that $pc1$ would have at least 6 SCSI devices and this is in contradiction with requirement $req5$. Therefore the configuration process has to backtrack and to revise some choices. It is worth noting that it would be useless to find an alternative configuration for the motherboard, since $mb1$ was configured while considering the subproblem relevant to $S_{11}$, which was independent from the one entailed by $S_{12}$ (for which the failure occurred). Therefore, let's suppose that the backtracking mechanism changes from SCSI to EIDE the types of the CD reader and of the CD writer playing the partonomic roles $has_cdr1$ and $has_cdw1$, respectively. After that, the tentative configuration $T4$ is produced (fig. 4.e). It is easy to see that $T4$ satisfies all the constraints in $S_1 = [req2, \ldots, req5, co1]$, therefore it represents a solution to the first of the two subproblems the main configuration problem $CP_0$ was decomposed into (see above).

To solve the main problem, the tentative configuration $T4$ must be extended in order to solve the subproblem entailed by $S_2 = [req1]$ too. $T5$ in fig. 4.f is a global solution.

This simple example illustrates a situation in which the configurator succeeds in further decomposing the current subproblem, after having inserted the direct components of the target object which the current set $currentSP$ of constraints refer to. However, it is worth noting that, in general, the configurator attempts to further decompose the current subproblem also after having completely configured each direct complex component of the target object (see the algorithm in fig. 3). Moreover, for the sake of simplicity, the example focuses only on the problem decomposition performed by partitioning the constraints relevant to the target object: it should be noticed that the decomposition is not limited to the target object, but, on the opposite, it is recursively performed also when configuring its complex (sub)components (by the execution of the recursive invocation in row 14) .

## 5 Experimental Results

The algorithm described in the previous section has been implemented in a configuration system written in Java (JDK 1.3). In this section we report some results from tests conducted with a computer system configuration domain. The experiments are aimed at testing the performance of the configuration algorithm

described in this paper and at comparing it (w.r.t. the computational effort) with a configuration strategy without decomposition and with the most performant decomposition strategy previously defined is this framework, the one called in [9] "strategy 3" (see [8, 9]). We call the algorithm in [9] *static decomposition* algorithm and the algorithm in Section 4.2 *dynamic decomposition* algorithm. All experiments were performed on a Mobile Pentium III 933 MHz 256 MB Windows 2000 notebook.

Using the computer system model, we generated *a test set of* 200 *configuration problems*; for each of them we specified the type of the target object (e.g. a PC for graphical applications) and some requirements that must be satisfied (e.g. it must have a CD writer of a certain kind, it must be *fast enough* and so on). In 83 problems we intentionally imposed a set of requirements inconsistent with the conceptual model (in average, these problems are quite hard). A problem is considered solved iff the configurator provides a solution or it detects that the problem does not admit any solution. For each problem the CPU time and the number of backtrackings that it required have been measured. The configuration algorithms include some random choices: e.g. after decomposing a problem, the selection of the current subproblem (see Section 4.2) is performed randomly. To reduce the bias due to "lucky" or "unlucky" random choices, every experiment was performed *ten times* and the average values of measured parameters were considered.

The strategy with dynamic decomposition proves to be effective in reducing the time and the number of backtrackings required by a problem to be solved w.r.t. both the algorithm without decomposition and the algorithm with static decomposition. Figure 5 shows the frequency histograms of the CPU times. On the X axis is reported the time interval taken in consideration and on Y axis is reported the number of problems solved within the given interval. The chart shows that the dynamic decomposition is rather effective in "moving" CPU times to low values, particularly to values less than 3 seconds. Figure 6 reports the relative cumulative distribution graphs for CPU times. In this case the Y axis reports

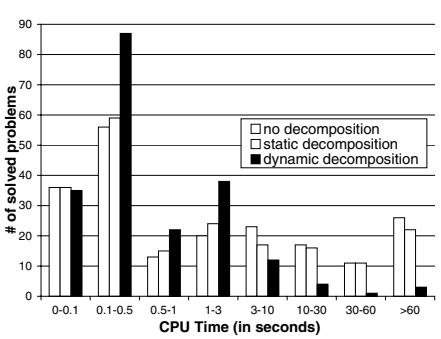



**Fig. 5.** Frequency histogram of CPU time

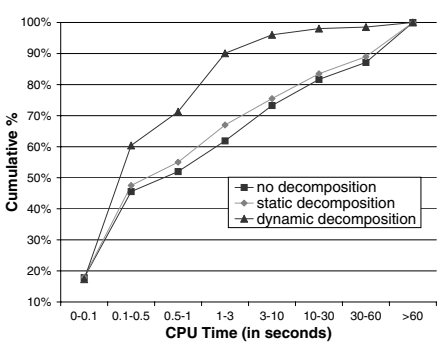



**Fig. 6.** Relative cumulative frequency graph of CPU Time

the cumulative frequencies of problems solved within the given interval. It may be worth to notice that the $90^{th}$ percentile for strategy without decomposition is 164 s, for static decomposition it is 68 s, while it is 2.5 s for strategy with dynamic decomposition. Results regarding CPU times are reflected by those regarding the number of backtrackings. Histograms and graphs are similar to those reported for CPU times (because of space constraints it is not possible to show them here). The $90^{th}$ percentile for the number of backtrackings is 14293 for no decomposition, 8056 for static decomposition and 323 for dynamic decomposition, resulting in a significative reduction of the number of backtrackings, too.

## 6 Conclusion and Discussion

In some configuration domains the theoretical intractability of configuration problems can appear also in practice since a few configuration problems can require a huge amount of CPU time to be solved. Some tasks, such as interactive configuration and on-line configuration on the Web, need low response times by the configurator, therefore the issue of controlling in practice the computational complexity of configuration problems should be dealt with.

In this paper we have investigated the role of problem decomposition in improving the efficiency of configurators.

Other researchers have recognized the importance of decomposition in solving difficult configuration problems. In particular, in [2], the authors stress the need of designing modular configuration models with low interaction among modules in such a way that the modules can be solved one by one. However, little attention has been paid to provide the configurator with mechanisms to automatically decompose configuration problems.

We have defined a decomposition technique, in a structured logical approach to configuration, that exploits compositional knowledge in order to partition configuration problems into a set of simpler (and independent) subproblems.

In [9, 8] some decomposition mechanisms were presented. Although these decomposition techniques have proved to be useful in reducing CPU times, still they do not allow to solve the large majority of the problems in a time acceptable for interactive and on-line configuration, i.e. in less than few seconds. In this work we have extended both the one called in [8] *constraints-splitting decomposition* and those defined in [9]. Differently from *constraints-splitting decomposition*, the mechanism presented here allows the configurator to perform decomposition recursively by partitioning both the constraints directly associated with the target object and those associated with its components and subcomponents. Moreover, in the decomposition techniques defined in [9, 8], the constraints are treated as static entities, while here we have proposed an improved mechanism that is able to perform more decompositions by dynamically simplifying the constraints during the configuration process.

Some experimental results conducted in a computer system configuration domain are reported which show the effectiveness of the decomposition technique presented here.

Few cases of the test set still required a huge amount of CPU time (more than 60 s), therefore we do not claim that decomposition is the "silver bullet" for difficult configuration problems. However, the experimental results suggest that it can play an important role in increasing the efficiency of configurators, therefore it is worth investigating various integrations of decomposition and other techniques (off-line knowledge compilation, re-using past solutions and so on).

## References


[1] G. Fleischanderl, G. E. Friedrich, A. Haselböck, H. Schreiner, and M. Stumptner. Configuring large systems using generative constraint satisfaction. *IEEE Intelligent Systems*, (July/August 1998):59–68, 1998. 39

[2] G. Fleischanderl and A. Haselböck. Thoughts on partitioning large-scale configuration problems. In *AAAI 1996 Fall Symposium Series*, pages 1–10, 1996. 50

[3] G. Friedrich and M. Stumptner. Consistency-based configuration. In *AAAI-99, Workshop on Configuration*, 1999. 39

[4] L. Geneste and M. Ruet. Fuzzy case based configuration. In *Proc. ECAI 2002 Configuration WS*, pages 71–76, 2002. 40

[5] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977. 39

[6] D. Magro and P. Torasso. Interactive configuration capability in a sale support system: Laziness and focusing mechanisms. In *Proc. IJCAI-01 Configuration WS*, pages 57–63, 2001. 40

[7] D. Magro and P. Torasso. Supporting product configuration in a virtual store. *LNAI*, 2175:176–188, 2001. 40, 41

[8] D. Magro and P. Torasso. Decomposition strategies for configuration problems. *AIEDAM, Special Issue on Configuration*, 17(1), 2003. 40, 43, 49, 50

[9] D. Magro, P. Torasso, and Luca Anselma. Problem decomposition in configuration. In *Proc. ECAI 2002 Configuration WS*, pages 50–55, 2002. 40, 43, 49, 50

[10] J. McDermott. R1: A rule-based configurer of computer systems. *Artificial Intelligence*, (19):39–88, 1982. 39

[11] D. L. McGuinness and J. R. Wright. An industrial-strength description logic-based configurator platform. *IEEE Intelligent Systems*, (July/August 1998):69–77, 1998. 39

[12] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proc. of the AAAI 90*, pages 25–32, 1990. 39

[13] D. Sabin and E. C. Freuder. Configuration as composite constraint satisfaction. In *Proc. Artificial Intelligence and Manufacturing. Research Planning Workshop*, pages 153–161, 1996. 39

[14] C. Sinz. Knowledge compilation for product configuration. In *Proc. ECAI 2002 Configuration WS*, pages 23–26, 2002. 40

[15] T. Soininen, E. Gelle, and I. Niemelä. A fixpoint definition of dynamic constraint satisfaction. In *LNCS 1713*, pages 419–433, 1999. 39

[16] T. Soininen, I. Niemelä, J. Tiihonen, and R. Sulonen. Representing configuration knowledge with weight constraint rules. In *Proc. of the AAAI Spring 2001 Symposium on Answer Set Programming*, 2001. 39

[17] J. Tiihonen, T. Soininen, I. Niemelä, and R. Sulonen. Empirical testing of a weight constraint rule based configurator. In *Proc. ECAI 2002 Configuration WS*, pages 17–22, 2002. 39

[18] M. Veron and M. Aldanondo. Yet another approach to ccsp for configuration problem. In *Proc. ECAI 2000 Configuration WS*, pages 59–62, 2000. 39

# Bridging the Gap between Horn Clausal Logic and Description Logics in Inductive Learning

Francesca A. Lisi and Donato Malerba

Dipartimento di Informatica, University of Bari, Italy
{lisi,malerba}@di.uniba.it

**Abstract.** This paper deals with spaces of inductive hypotheses represented with hybrid languages. We adopt $\mathcal{AL}$-log, a language that combines the function-free Horn clausal language Datalog and the description logic $\mathcal{ALC}$ by using concept assertions as type constraints on variables. For constrained Datalog clauses we define a relation of subsumption, called $\mathcal{B}$-subsumption, inspired by Buntine's generalized subsumption. We show that $\mathcal{B}$-subsumption induces a quasi-order, denoted as $\succeq_{\mathcal{B}}$, over the space of constrained Datalog clauses and provide a procedure for checking $\succeq_{\mathcal{B}}$ under the object identity bias.

## 1 Introduction

Many problems in concept learning and data mining require a search process through a partially ordered space of hypotheses. E.g., in Mannila's approach to frequent pattern discovery [14], the space of patterns is organized according to a generality order $\succeq$ between patterns and searched one level at a time, starting from the most general patterns and iterating between candidate generation and candidate evaluation phases. The idea of taking advantage of partial orders to efficiently search a space of inductive hypotheses dates back to Mitchell's work on concept learning [15] and had a great influence on studies of concept learning in Horn clausal logic, known under the name of Inductive Logic Programming (ILP) [16]. E.g., [14] has been shown to be related to *characteristic induction*, i.e. the logical framework for induction in ILP which is particularly suitable for tasks of description [8].

In the 'generalization as search' approach the space of inductive hypotheses is strongly biased by the hypothesis representation. Indeed, by selecting a hypothesis representation, the designer of the learning/mining algorithm implicitly defines the space of all hypotheses that the program can ever represent and therefore can ever learn/mine. Recently there has been a growing interest in application areas, e.g. the Semantic Web, which demand for learning/mining in highly expressive representation languages. Description logics (DLs) are considered especially effective for those domains where the knowledge can be easily organized along is-a hierarchies [1]. The ability to represent and reason about taxonomies in DLs has motivated their use as a modeling language in the design and maintenance of large, hierarchically structured bodies of knowledge. This

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 53–65, 2003.
© Springer-Verlag Berlin Heidelberg 2003

work has been more or less explicitly applied to inductive learning [5, 6]. People from the ILP community also has become interested in learning DLs [10, 2]. All these proposals deal with pure DL formalisms. Yet hybrid languages, such as CARIN [11] and $\mathcal{AL}$-log [7], seem more interesting because they combine Horn clausal logic and description logics. The work presented in [18] is the first attempt at learning in hybrid languages. Here, the chosen language is CARIN-$\mathcal{ALN}$ and algorithms for testing example coverage and subsumption between two hypotheses are based on the existential entailment algorithm. In this paper we deal with spaces of inductive hypotheses represented with $\mathcal{AL}$-log. This language merges DATALOG [4] and $\mathcal{ALC}$ [19] by using concept assertions essentially as *type constraints* on variables. We define $\mathcal{B}$-subsumption, a generality relation for constrained Datalog clauses, and illustrate its application in inductive problems by means of an example of frequent pattern discovery in the aforementioned Mannila's formulation.

The paper is organized as follows. Section 2 introduces the hybrid language $\mathcal{AL}$-log. Section 3 is devoted to the investigation of subsumption between constrained DATALOG clauses. Section 4 presents an application of $\mathcal{B}$-subsumption to frequent pattern discovery problems. Section 5 concludes the paper with final remarks.

## 2 $\mathcal{AL}$-log=$\mathcal{ALC}$+Datalog

The language $\mathcal{AL}$-log combines the representation and reasoning means offered by DATALOG and $\mathcal{ALC}$. Indeed it embodies two subsystems, called *relational* and *structural*. We assume the reader to be familiar with DATALOG, therefore we focus on the structural subsystem and hybridization of the relational subsystem.

### 2.1 The $\mathcal{ALC}$ Component

The structural subsystem of $\mathcal{AL}$-log allows for the specification of structural knowledge in terms of *concepts*, *roles*, and *individuals*. Individuals represent objects in the domain of interest. Concepts represent classes of these objects, while roles represent binary relations between concepts. Complex concepts can be defined by means of constructs, such as $\sqcap$ and $\sqcup$. The structural subsystem is itself a two-component system. The *intensional* component $\mathcal{T}$ consists of concept hierarchies spanned by is-a relations between concepts, namely *inclusion statements* of the form $C \sqsubseteq D$ (read "$C$ is included in $D$") where $C$ and $D$ are two arbitrary concepts. The *extensional* component $\mathcal{M}$ specifies instance-of relations, e.g. *concept assertions* of the form $a : C$ (read "$a$ belongs to $C$") where $a$ is an individual and $C$ is a concept.

In $\mathcal{ALC}$ knowledge bases, an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$). E.g., it maps concepts to subsets of $\Delta^{\mathcal{I}}$ and individuals to elements of $\Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (see *unique names* assumption [17]). We say that $\mathcal{I}$ is a *model* for $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and for $a : C$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$.

The main reasoning mechanism for the structural component is the satisfiability check. The *tableau calculus* proposed in [7] starts with the tableau branch $S = \mathcal{T} \cup \mathcal{M}$ and adds assertions to $S$ by means of *propagation rules* such as

- $S \rightarrow_{\sqcup} S \cup \{s : D\}$ if
  1. $s : C_1 \sqcup C_2$ is in $S$,
  2. $D = C_1$ and $D = C_2$,
  3. neither $s : C_1$ nor $s : C_2$ is in $S$
- $S \rightarrow_{\sqsubseteq} S \cup \{s : C' \sqcup D\}$ if
  1. $C \sqsubseteq D$ is in $S$,
  2. $s$ appears in $S$,
  3. $C'$ is the NNF concept equivalent to $\neg C$
  4. $s : \neg C \sqcup D$ is not in $S$
- $S \rightarrow_{\perp} \{s : \perp\}$ if
  1. $s : A$ and $s : \neg A$ are in $S$, or
  2. $s : \neg\top$ is in $S$,
  3. $s : \perp$ is not in $S$

until either a contradiction is generated or an interpretation satisfying $S$ can be easily obtained from it.

### 2.2 Hybridization of Datalog

In $\mathcal{AL}$-log one can define Datalog programs enriched with *constraints* of the form $s : C$ where $s$ is either a constant or a variable, and $C$ is an $\mathcal{ALC}$-concept. Note that the usage of concepts as typing constraints applies only to variables and constants that already appear in the clause. The symbol & separates constraints from Datalog atoms in a clause.

**Definition 1.** *A* constrained Datalog clause *is an implication of the form* $\alpha_0 \leftarrow \alpha_1, \ldots, \alpha_m \& \gamma_1, \ldots, \gamma_n$ *where* $m \geq 0$, $n \geq 0$, $\alpha_i$ *are* Datalog *atoms and* $\gamma_j$ *are constraints. A* constrained Datalog program $\Pi$ *is a set of constrained* Datalog *clauses.*

An $\mathcal{AL}$*-log knowledge base* $\mathcal{B}$ is the pair $\langle \Sigma, \Pi \rangle$ where $\Sigma$ is an $\mathcal{ALC}$ knowledge base and $\Pi$ is a constrained Datalog program. For a knowledge base to be acceptable, it must satisfy the following conditions:

- The set of Datalog predicate symbols appearing in $\Pi$ is disjoint from the set of concept and role symbols appearing in $\Sigma$.
- The alphabet of constants in $\Pi$ coincides with the alphabet $\mathcal{O}$ of the individuals in $\Sigma$. Also, every constant occurring in $\Pi$ appears also in $\Sigma$.
- For every clause in $\Pi$, every variable occurring in the constraint part occurs also in the Datalog part.

These properties allow for the extension of terminology and results related to the notion of *substitution* from Datalog to $\mathcal{AL}$-log in a straightforward manner.

*Example 1.* As a running example, we consider an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ obtained from the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database distributed by Microsoft$^{TM}$ as sample database for MS Access. The structural subsystem $\Sigma$ should reflect the E/R model underlying the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database. To serve our illustrative purpose we focus on the concepts (entities) `Order` and `Product`. The intensional part of $\Sigma$ contains assertions such as `DairyProduct` $\sqsubset$ `Product` that define a taxonomy on products w.r.t. their category. The extensional part of $\Sigma$ contains 830 concept assertions for `Order` (e.g. `order10248:Order`), and 77 assertions for the sub-concepts of `Product`, e.g. `product11:DairyProduct`. The relational subsystem $\Pi$ expresses the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database as a constrained DATALOG program. The extensional part of $\Pi$ encompasses 2155 facts for `orderDetail`/5, e.g. `orderDetail(order10248,product11,'£14',12,0.00)` represents the order detail concerning the order number 10248 and product code 11. The intensional part of $\Pi$ defines a view on `orderDetail`:

```
item(OrderID,ProductID)← orderDetail(OrderID,ProductID,_,_,_)
                         & OrderID:Order, ProductID:Product
```

This rule when triggered on the EDB of $\Pi$ makes implicit facts explicit, such as `item(order10248,product11)`.

The interaction between the structural and the relational part of an $\mathcal{AL}$-log knowledge base is also at the basis of a model-theoretic semantics for $\mathcal{AL}$-log. We call $\Pi_D$ the set of DATALOG clauses obtained from the clauses of $\Pi$ by deleting their constraints. We define an *interpretation* $\mathcal{J}$ for $\mathcal{B}$ as the union of an $\mathcal{O}$-interpretation $\mathcal{I}_\mathcal{O}$ for $\Sigma$ (i.e. an interpretation compliant with the unique names assumption) and an Herbrand interpretation $\mathcal{I}_\mathcal{H}$ for $\Pi_D$. An interpretation $\mathcal{J}$ is a *model* of $\mathcal{B}$ if $\mathcal{I}_\mathcal{O}$ is a model of $\Sigma$, and for each ground instance $\bar{\alpha}'\&\gamma'_1,\ldots,\gamma'_n$ of each clause $\bar{\alpha}\&\gamma_1,\ldots,\gamma_n$ in $\Pi$, either there exists one $\gamma'_i$, $i \in \{1,\ldots,n\}$, that is not satisfied by $\mathcal{J}$, or $\bar{\alpha}'$ is satisfied by $\mathcal{J}$. The notion of *logical consequence* paves the way to the definition of answer set for queries. *Queries* to $\mathcal{AL}$-log knowledge bases are special cases of Definition 1. Since a query is an existentially quantified conjunction of atoms and constraints we have:

**Definition 2.** *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base. An* answer *to the query $Q$ is a ground substitution $\sigma$ for the variables in $Q$. The answer $\sigma$ is* correct *w.r.t. $\mathcal{B}$ if $Q\sigma$ is a logical consequence of $\mathcal{B}$ ($\mathcal{B} \models Q\sigma$). The* answer set *of $Q$ in $\mathcal{B}$, denoted as answerset$(Q,\mathcal{B})$, contains all the correct answers to $Q$ w.r.t. $\mathcal{B}$.*

The main reasoning service for $\mathcal{AL}$-log knowledge bases is *hybrid deduction* which is based on constrained SLD-resolution.

**Definition 3.** *Let $Q$ be a query $\leftarrow \beta_1,\ldots,\beta_m \ \&\gamma_1,\ldots,\gamma_n$, $E$ a constrained* DATALOG *clause $\alpha_0 \leftarrow \alpha_1,\ldots,\alpha_m\&\xi_1,\ldots,\xi_h$, and $\theta$ the most general substitution such that $\alpha_0\theta = \beta_j\theta$ where $\beta_j \in \{\beta_1,\ldots,\beta_m\}$. The* resolvent *of $Q$ and $E$ with substitution $\theta$ is the query $Q'$ having $(\beta_1,\ldots,\beta_{j-1},\alpha_1,\ldots,\alpha_m,\beta_{j+1},\ldots,\beta_m)\theta$ as* DATALOG *part, and $\gamma'_1,\ldots,\gamma'_k$ as constraints obtained from $\gamma_1\theta,\ldots,\gamma_n\theta,\xi_1,\ldots,\xi_h$ by applying the following simplifications: couples of constraints $t : C$, $t : D$ are replaced by the equivalent constraint $t : C \sqcap D$.*

The one-to-one mapping between constrained SLD-derivations and the SLD-derivations obtained by ignoring the constraints is exploited to extend known results for DATALOG to $\mathcal{AL}$-log. Note that in $\mathcal{AL}$-log a derivation of the empty clause with associated constraints does not represent a refutation. It actually infers that the query is true in those models of $\mathcal{B}$ that satisfy its constraints. This is due to the open-world assumption according to which an $\mathcal{ALC}$ knowledge base (in particular, the assertional part) represents possibly infinitely many interpretations, namely its models, as opposed to databases in which the closed-world assumption holds. Therefore in order to answer a query it is necessary to collect enough derivations ending with a constrained empty clause such that every model of $\mathcal{B}$ satisfies the constraints associated with the final query of at least one derivation. Formally:

**Definition 4.** *Let $Q^{(0)}$ be a query $\leftarrow \beta_1, \ldots, \beta_m \& \gamma_1, \ldots, \gamma_n$ to a $\mathcal{AL}$-log knowledge base $\mathcal{B}$. A* constrained SLD-refutation *for $Q^{(0)}$ in $\mathcal{B}$ is a finite set $\{d_1, \ldots, d_m\}$ of constrained SLD-derivations for $Q^{(0)}$ in $\mathcal{B}$ such that:*

1. *for each derivation $d_i$, $i \in \{1, \ldots, m\}$, the last query $Q^{(n_i)}$ of $d_i$ is a constrained empty clause;*
2. *for every model $\mathcal{J}$ of $\mathcal{B}$, there exists at least one derivation $d_i$, $i \in \{1, \ldots, m\}$, such that $\mathcal{J} \models Q^{(n_i)}$*

**Definition 5.** *Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base. An answer $\sigma$ to a query $Q$ is called a* computed answer *if there exists a constrained SLD-refutation for $Q\sigma$ in $\mathcal{B}$ ($\mathcal{B} \vdash Q\sigma$). The set of computed answers is called the* success set *of $Q$ in $\mathcal{B}$.*

**Lemma 1.** *[7] Let $Q$ be a ground query to a $\mathcal{AL}$-log knowledge base $\mathcal{B}$. It holds that $\mathcal{B} \vdash Q$ if and only if $\mathcal{B} \models Q$.*

Given *any* query $Q$, the success set of $Q$ in $\mathcal{B}$ coincides with the answer set of $Q$ in $\mathcal{B}$. This provides an operational means for computing correct answers to queries. Indeed, it is straightforward to see that the usual reasoning methods for DATALOG allow us to collect in a finite number of steps (actually in a number of steps which is polynomial with respect to the size of the extensional component of the constrained DATALOG program) enough constrained SLD-derivations for $Q$ in $\mathcal{B}$ to construct a refutation - if any. Derivations must satisfy both conditions of Definition 4. In particular, the latter requires some reasoning on the structural component of $\mathcal{B}$ as shown below.

*Example 2.* Following Example 1, we compute a correct answer to

$Q = \leftarrow$ `item(order10248,Y)` & `order10248:Order`, `Y:DairyProduct`

w.r.t. $\mathcal{B}$. Several refutations can be constructed for $Q = Q^{(0)}$. One of them consists of the following single constrained SLD-derivation.

Let $E^{(1)}$ be

```
item(OrderID,ProductID) ← orderDetail(OrderID,ProductID,_,_,_)
                          & OrderID:Order, ProductID:Product
```

A resolvent for $Q^{(0)}$ and $E^{(1)}$ with substitution $\sigma^{(1)} = \{$`OrderID/ order10248`, `ProductID/ Y`$\}$ is the query

$$Q^{(1)} = \leftarrow \texttt{orderDetail(order10248,Y,_,_,_)} \\ \& \ \texttt{order10248:Order}, \texttt{Y:DairyProduct}$$

Let $E^{(2)}$ be `orderDetail(order10248,product11,'£14',12,0.00)`. A resolvent for $Q^{(1)}$ and $E^{(2)}$ with substitution $\sigma^{(2)} = \{$`Y/ product11`$\}$ is the constrained empty clause

$$Q^{(2)} = \leftarrow \& \ \texttt{order10248:Order}, \texttt{product11:DairyProduct}$$

What we need to check is that $\Sigma \cup \{$`order10248:Order`, `product11:DairyProduct`$\}$ is satisfiable. This check amounts to two unsatisfiability checks to be performed by applying the tableau calculus. The first check operates on the initial tableau $S^{(0)} = \Sigma \cup \{\texttt{order10248:}\neg\texttt{Order}\}$. The application of the propagation rule $\rightarrow_{\perp}$ to $S^{(0)}$ produces the tableau $S^{(1)} = \{\texttt{order10248:}\perp\}$. Computation stops here because no other rule can be applied to $S^{(1)}$. Since $S^{(1)}$ is complete and contains a clash, the initial tableau $S^{(0)}$ is unsatisfiable. The second check operates on the initial tableau $S^{(0)} = \Sigma \cup \{\texttt{product11:}\neg\texttt{DairyProduct}\}$. It also terminates with a clash after applying $\rightarrow_{\perp}$ to $S^{(0)}$. These two results together prove the satisfiability of $\Sigma \cup \{$`order10248:Order`, `product11:DairyProduct`$\}$, then the correcteness of $\sigma$=$\{$`Y/ product11`$\}$ as an answer to $Q$ w.r.t. $\mathcal{B}$.

## 3 The $\mathcal{B}$-subsumption Relation

The definition of a subsumption relation for constrained Datalog clauses can disregard neither the peculiarities of $\mathcal{AL}$-log nor the methodological apparatus of ILP. This results in some adjustments of $\mathcal{AL}$-log in order to make it a knowledge representation and reasoning framework suitable for our purposes.

First, we impose constrained Datalog clauses to be linked and connected (or range-restricted). Linkedness has been originally conceived for definite clauses [9]. Connectedness is the ILP counterpart of the safety condition that any Datalog clause must satisfy [4].

**Definition 6.** *Let $C$ be a constrained* Datalog *clause. A term $t$ in some literal $l_i \in C$ is* linked *with linking-chain of length 0, if $t$ occurs in head($C$), and is linked with linking-chain of length $d+1$, if some other term in $l_i$ is linked with linking-chain of length $d$. The link-depth of a term $t$ in some $l_i \in C$ is the length of the shortest linking-chain of $t$. A literal $l_i \in C$ is linked if at least one of its terms is linked. The clause $C$ itself is linked if each $l_i \in C$ is linked. The clause $C$ is* connected *if each variable occurring in head($C$) also occur in body($C$).*

Second, we extend the unique names assumption from the semantic level to the syntactic one. Note that the unique names assumption holds naturally for ground constrained Datalog clauses because the semantics of $\mathcal{AL}$-log adopts

Herbrand models for the DATALOG part and $\mathcal{O}$-models for the constraint part. Conversely it is not guaranteed in the case of non-ground constrained DATALOG clauses, e.g. different variables can be unified. We propose to impose the bias of Object Identity [20] on the $\mathcal{AL}$-log framework: In a formula, terms denoted with different symbols must be distinct, i.e. they represent different entities of the domain. This bias can be the starting point for the definition of either an equational theory or a quasi-ordering for constrained DATALOG clauses. The latter option relies on a restricted form of substitution whose bindings avoid the identification of terms: A substitution $\sigma$ is an *OI-substitution* w.r.t. a set of terms $T$ iff $\forall t_1, t_2 \in T\colon\ t_1 \neq t_2$ yields that $t_1\sigma \neq t_2\sigma$. From now on, we assume that substitutions are OI-compliant. See [12] for an investigation of OI in the case of DATALOG queries.

Third, we rely on the reasoning mechanisms made available by $\mathcal{AL}$-log knowledge bases. Generalized subsumption [3] has been introduced in ILP as a generality order for Horn clauses with respect to background knowledge. We propose to adapt it to our $\mathcal{AL}$-log framework as follows.

**Definition 7.** *Let $Q$ be a constrained* DATALOG *clause, $\alpha$ a ground atom, and $\mathcal{J}$ an interpretation. We say that $Q$* covers *$\alpha$ under $\mathcal{J}$ if there is a ground substitution $\theta$ for $Q$ ($Q\theta$ is ground) such that $body(Q)\theta$ is true under $\mathcal{J}$ and $head(Q)\theta = \alpha$.*

**Definition 8.** *Let $P$, $Q$ be two constrained* DATALOG *clauses and $\mathcal{B}$ an $\mathcal{AL}$-log knowledge base. We say that $P$ $\mathcal{B}$-*subsumes *$Q$ if for every model $\mathcal{J}$ of $\mathcal{B}$ and every ground atom $\alpha$ such that $Q$ covers $\alpha$ under $\mathcal{J}$, we have that $P$ covers $\alpha$ under $\mathcal{J}$.*

We can define a generality relation $\succeq_{\mathcal{B}}$ for constrained DATALOG clauses on the basis of $\mathcal{B}$-subsumption. It can be easily proven that $\succeq_{\mathcal{B}}$ is a quasi-order (i.e. it is a reflexive and transitive relation) for constrained DATALOG clauses.

**Definition 9.** *Let $P$, $Q$ be two constrained* DATALOG *clauses and $\mathcal{B}$ an $\mathcal{AL}$-log knowledge base. We say that $P$ is* at least as general as *$Q$ under $\mathcal{B}$-subsumption, $P \succeq_{\mathcal{B}} Q$, iff $P$ $\mathcal{B}$-subsumes $Q$. Furthermore, $P$ is* more general than *$Q$ under $\mathcal{B}$-subsumption, $P \succ_{\mathcal{B}} Q$, iff $P \succeq_{\mathcal{B}} Q$ and $Q \not\succeq_{\mathcal{B}} P$. Finally, $P$ is* equivalent *to $Q$ under $\mathcal{B}$-subsumption, $P \sim_{\mathcal{B}} Q$, iff $P \succeq_{\mathcal{B}} Q$ and $Q \succeq_{\mathcal{B}} P$.*

The next two lemmas show the definition of $\mathcal{B}$-subsumption to be equivalent to another formulation, which will be more convenient in later proofs than the definition based on covering.

**Definition 10.** *Let $\mathcal{B}$ be an $\mathcal{AL}$-log knowledge base and $H$ be a constrained* DATALOG *clause. Let $X_1, \ldots, X_n$ be all the variables appearing in $H$, and $a_1, \ldots, a_n$ be distinct constants (individuals) not appearing in $\mathcal{B}$ or $H$. Then the substitution $\{X_1/a_1, \ldots, X_n/a_n\}$ is called a* Skolem substitution *for $H$ w.r.t. $\mathcal{B}$.*

**Lemma 2.** *Let $P$, $Q$ be two constrained* DATALOG *clauses, $\mathcal{B}$ an $\mathcal{AL}$-log knowledge base, and $\sigma$ a Skolem substitution for $Q$ with respect to $\{P\} \cup \mathcal{B}$. We*

*say that* $P \succeq_{\mathcal{B}} Q$ *iff there exists a ground substitution* $\theta$ *for* $P$ *such that (i)* $head(P)\theta = head(Q)\sigma$ *and (ii)* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta$.

*Proof.* ($\Rightarrow$) *Suppose* $P \succeq_{\mathcal{B}} Q$. *Let* $\mathcal{B}'$ *be the knowledge base* $\mathcal{B} \cup body(Q)\sigma$ *and* $\mathcal{J} = \langle \mathcal{I}_{\mathcal{O}}, \mathcal{I}_H \rangle$ *be a model of* $\mathcal{B}'$ *where* $\mathcal{I}_{\mathcal{O}}$ *is the minimal* $\mathcal{O}$*-model of* $\Sigma$ *and* $\mathcal{I}_H$ *be the least Herbrand model of the* DATALOG *part of* $\mathcal{B}'$. *The substitution* $\sigma$ *is a ground substitution for* $Q$, *and* $body(Q)\sigma$ *is true under* $\mathcal{J}$, *so* $Q$ *covers* $head(Q)\sigma$ *under* $\mathcal{J}$ *by Definition 7. Then* $P$ *must also cover* $head(Q)\sigma$ *under* $\mathcal{J}$. *Thus there is a ground substitution* $\theta$ *for* $P$ *such that* $head(P)\theta = head(Q)\sigma$, *and* $body(P)\theta$ *is true under* $\mathcal{J}$, *i.e.* $\mathcal{J} \models body(P)\theta$. *By properties of the least Herbrand model, it holds that* $\mathcal{B} \cup body(Q)\sigma \models \mathcal{J}$, *hence* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta$.

($\Leftarrow$) *Suppose there is a ground substitution* $\theta$ *for* $P$, *such that* $head(P)\theta = head(Q)\sigma$ *and* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta$. *Let* $\alpha$ *be some ground atom and* $\mathcal{J}_\alpha$ *some model of* $\mathcal{B}$ *such that* $Q$ *covers* $\alpha$ *under* $\mathcal{J}_\alpha$. *To prove that* $P \succeq_{\mathcal{B}} Q$ *we need to prove that* $P$ *covers* $\alpha$ *under* $\mathcal{J}_\alpha$.

*Construct a substitution* $\theta'$ *from* $\theta$ *as follows: for every binding* $X/c \in \sigma$, *replace* $c$ *in bindings in* $\theta$ *by* $X$. *Then we have* $P\theta'\sigma = P\theta$ *and none of the Skolem constants of* $\sigma$ *occurs in* $\theta'$. *Then* $head(P)\theta'\sigma = head(P)\theta = head(Q)\sigma$, *so* $head(P)\theta' = head(Q)$. *Since* $Q$ *covers* $\alpha$ *under* $\mathcal{J}_\alpha$, *there is a ground substitution* $\gamma$ *for* $Q$, *such that* $body(Q)\gamma$ *is true under* $\mathcal{J}_\alpha$, *and* $head(Q)\gamma = \alpha$. *This implies that* $head(P)\theta'\gamma = head(Q)\gamma = \alpha$.

*It remains to show that* $body(P)\theta'\gamma$ *is true under* $\mathcal{J}_\alpha$. *Since* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta'\sigma$ *and* $\leftarrow body(P)\theta'\sigma$ *is a ground query, it follows from Lemma 1 that there exists a constrained SLD-refutation for* $\leftarrow body(P)\theta'\sigma$ *in* $\mathcal{B} \cup body(Q)\sigma$. *By Definition 4 there exists a finite set* $\{d_1, \ldots, d_m\}$ *of constrained SLD-derivations, having* $\leftarrow body(P)\theta'\sigma$ *as top clause and elements of* $\mathcal{B} \cup body(Q)\sigma$ *as input clauses, such that for each derivation* $d_i$, $i \in \{1, \ldots, m\}$, *the last query* $Q^{(n_i)}$ *of* $d_i$ *is a constrained empty clause and for every model* $\mathcal{J}$ *of* $\mathcal{B} \cup body(Q)\sigma$, *there exists at least one derivation* $d_i$, $i \in \{1, \ldots, m\}$, *such that* $\mathcal{J} \models Q^{(n_i)}$. *We want to turn this constrained SLD-refutation for* $\leftarrow body(P)\theta'\sigma$ *in* $\mathcal{B} \cup body(Q)\sigma$ *into a constrained SLD-refutation for* $\leftarrow body(P)\theta'\gamma$ *in* $\mathcal{B} \cup body(Q)\gamma$, *thus proving that* $\mathcal{B} \cup body(Q)\gamma \models body(P)\theta'\gamma$. *Let* $X_1, \ldots, X_n$ *be the variables in* $body(Q)$, $\{X_1/c_1, \ldots, X_n/c_n\} \subseteq \sigma$, *and* $\{X_1/t_1, \ldots, X_n/t_n\} \subseteq \gamma$. *If we replace each Skolem constant* $c_j$ *by* $t_j$, $1 \leq j \leq n$, *in both the SLD-derivations and the models of* $\mathcal{B} \cup body(Q)\sigma$ *we obtain a constrained SLD-refutation of* $body(P)\theta'\gamma$ *in* $\mathcal{B} \cup body(Q)\gamma$. *Hence* $\mathcal{B} \cup body(Q)\gamma \models body(P)\theta'\gamma$. *Since* $\mathcal{J}_\alpha$ *is a model of* $\mathcal{B} \cup body(Q)\gamma$, *it is also a model of* $body(P)\theta'\gamma$.

The relation between $\mathcal{B}$-subsumption and constrained SLD-resolution is given below. It provides an operational means for checking $\mathcal{B}$-subsumption.

**Theorem 1.** *Let* $P$, $Q$ *be two constrained* DATALOG *clauses,* $\mathcal{B}$ *an* $\mathcal{AL}$*-log knowledge base, and* $\sigma$ *a Skolem substitution for* $Q$ *with respect to* $\{P\} \cup \mathcal{B}$. *We say that* $P \succeq_{\mathcal{B}} Q$ *iff there exists a substitution* $\theta$ *for* $P$ *such that (i)* $head(P)\theta = head(Q)$ *and (ii)* $\mathcal{B} \cup body(Q)\sigma \vdash body(P)\theta\sigma$ *where* $body(P)\theta\sigma$ *is ground.*

*Proof. By Lemma 2, we have* $P \succeq_{\mathcal{B}} Q$ *iff there exists a ground substitution* $\theta'$ *for* $P$*, such that* $head(P)\theta' = head(Q)\sigma$ *and* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta'$*. Since* $\sigma$ *is a Skolem substitution, we can define a substitution* $\theta$ *such that* $P\theta\sigma = P\theta'$ *and none of the Skolem constants of* $\sigma$ *occurs in* $\theta$*. Then* $head(P)\theta = head(Q)$ *and* $\mathcal{B} \cup body(Q)\sigma \models body(P)\theta\sigma$*. Since* $body(P)\theta\sigma$ *is ground, by Lemma 1 we have* $\mathcal{B} \cup body(Q)\sigma \vdash body(P)\theta\sigma$*, so the thesis follows.*

The decidability of $\mathcal{B}$-subsumption follows from the decidability of both generalized subsumption in DATALOG [3] and query answering in $\mathcal{AL}$-log [7].

## 4 An Application to Frequent Pattern Discovery

The relation of $\mathcal{B}$-subsumption can be adopted for structuring the space of patterns in frequent pattern discovery problems. A *frequent pattern* is an intensional description of a subset of a given data set whose cardinality exceeds the user-defined threshold. The *frequent pattern discovery* is to generate *all* frequent patterns.

In our $\mathcal{AL}$-log framework for frequent pattern discovery, *data* is represented as an $\mathcal{AL}$-log knowledge base and *patterns* are to be intended as unary conjunctive queries called $\mathcal{O}$-queries. More precisely, given a $\mathcal{ALC}$ concept $\hat{C}$ of reference, an *$\mathcal{O}$-query* $Q$ to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is a constrained DATALOG clause of the form

$$Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma_2, \ldots, \gamma_n$$

where $X$ is the *distinguished variable* and the remaining variables occurring in the body of $Q$ are the *existential variables*. A *trivial* $\mathcal{O}$-query is a constrained empty clause of the form $q(X) \leftarrow \& X : \hat{C}$.

*Example 3.* The following $\mathcal{O}$-queries

$Q_0$= `q(X)` $\leftarrow$ & `X:Order`
$Q_1$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`
$Q_3$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`, `Y:Product`
$Q_5$= `q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`
$Q_9$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`, `Y:DairyProduct`

represent patterns describing the reference concept `Order` with respect to other concepts occurring in the $\mathcal{AL}$-log knowledge base sketched in Example 1.

The aforementioned conditions of linkedness and connectedness guarantee that the evaluation of $\mathcal{O}$-queries is sound. In particular, an answer $\theta$ to an $\mathcal{O}$-query $Q$ is a *correct (resp. computed) answer* w.r.t. $\mathcal{B}$ if there exists at least one correct (resp. computed) answer to $body(Q)\theta$ w.r.t. $\mathcal{B}$. Therefore the answer set of $Q$ will contain individuals of the reference concept $\hat{C}$. Its cardinality gives the absolute frequency of $Q$ in $\mathcal{B}$.

*Example 4.* Following Example 2 and 3, the substitution $\theta = \{$`X/order10248`$\}$ is a correct answer to $Q_9$ w.r.t. $\mathcal{B}$ because there exists a correct answer $\sigma$=$\{$`Y/product11`$\}$ to $body(Q_9)\theta$ w.r.t. $\mathcal{B}$.

There are three main advantages in adopting $\mathcal{B}$-subsumption as a generality order in frequent pattern discovery problems. First, the hybrid nature of $\mathcal{AL}$-log provides a unified treatment of both relational and structural features of data. Second, the OI bias make patterns compliant with the unique names assumption (see Example 5). Last, the underlying reasoning mechanisms of $\mathcal{AL}$-log enable the discovery of patterns at multiple description granularity levels. E.g., $Q_9$ is a finer-grained version of $Q_3$ with respect to the taxonomy on products reported in Example 1. This relation between the two queries is captured by $\mathcal{B}$-subsumption as illustrated in Example 6.

*Example 5.* Let us consider the $\mathcal{O}$-queries

$Q_3$= q(A) $\leftarrow$ item(A,B) & A:Order, B:Product
$Q_5$= q(X) $\leftarrow$ item(X,Y), item(X,Z) & X:Order, Y:Product

reported in Example 3 up to variable renaming. We want to check whether $Q_3 \succeq_{\mathcal{B}} Q_5$ holds. Let $\sigma$={X/a, Y/b, Z/c} a Skolem substitution for $Q_5$ with respect to $\mathcal{B} \cup \{Q_3\}$ and $\theta$={A/X, B/Y} a substitution for $Q_3$. The condition (i) is immediately verified. It remains to verify that

$$\text{(ii) } \mathcal{B} \cup \{\texttt{item(a,b)}, \texttt{item(a,c)\&a:Order}, \texttt{b:Product}\} \models \texttt{item(a,b)\&a:Order}, \texttt{b:Product}.$$

We try to build a constrained SLD-refutation for

$Q^{(0)} = \leftarrow$ item(a,b) & a:Order, b:Product

in $\mathcal{B}' = \mathcal{B} \cup \{\texttt{item(a,b)}, \texttt{item(a,c)\&a:Order}, \texttt{b:Product}\}$.

Let $E^{(1)}$ be item(a,b). A resolvent for $Q^{(0)}$ and $E^{(1)}$ with the empty substitution $\sigma^{(1)}$ is the constrained empty clause

$Q^{(1)} = \leftarrow$ & a:Order, b:Product

What we need to check is that $\Sigma' \cup \{\texttt{a:Order}, \texttt{b:Product}\}$ is satisfiable. The first unsatisfiability check operates on the initial tableau $S^{(0)} = \Sigma' \cup \{\texttt{a:}\neg\texttt{Order}\}$. The application of the propagation rule $\rightarrow_{\perp}$ to $S^{(0)}$ produces the tableau $S^{(1)} = \{\texttt{a:}\perp\}$. Computation stops here because no other rule can be applied to $S^{(1)}$. Since $S^{(1)}$ is complete and contains a clash, the initial tableau $S^{(0)}$ is unsatisfiable. The second unsatisfiability check operates on the initial tableau $S^{(0)} = \Sigma' \cup \{\texttt{b:}\neg\texttt{Product}\}$. By applying $\rightarrow_{\perp}$ to $S^{(0)}$ we obtain again a complete tableau with a clash.

These two results together prove the satisfiability of $\Sigma \cup \{$a:Order, b:Product$\}$, then the existence of a constrained SLD-refutation for $Q^{(0)}$ in $\mathcal{B}'$. Therefore we can say that $Q_3 \succeq_{\mathcal{B}} Q_5$. Note that $Q_5 \not\succeq_{\mathcal{B}} Q_3$ under the object identity bias. Indeed this bias does not admit the substitution $\theta$={X/A, Y/B, Z/B } for $Q_5$ which would make possible to verify conditions (i) and (ii) of Theorem 1.

*Example 6.* Let us consider the $\mathcal{O}$-queries

$Q_3$= q(A) $\leftarrow$ item(A,B) & A:Order, B:Product
$Q_9$= q(X) $\leftarrow$ item(X,Y) & X:Order, Y:DairyProduct

reported in Example 3 up to variable renaming. We want to check whether $Q_3 \succeq_{\mathcal{B}} Q_9$ holds. Let $\sigma$={X/a, Y/b} a Skolem substitution for $Q_9$ w.r.t. $\mathcal{B} \cup \{Q_3\}$ and $\theta$={A/X, B/Y} a substitution for $Q_3$. The condition (i) is immediately verified. It remains to verify that (ii) $\mathcal{B} \cup \{$item(a,b)&a:Order, b:DairyProduct$\} \models$ item(a,b) & a:Order, b:Product. We try to build a constrained SLD-refutation for

$$Q^{(0)} = \leftarrow \texttt{item(a,b)} \ \& \ \texttt{a:Order}, \texttt{b:Product}$$

in $\mathcal{B}' = \mathcal{B} \cup \{$item(a,b)&a:Order, b:DairyProduct$\}$.

Let $E^{(1)}$ be item(a,b). A resolvent for $Q^{(0)}$ and $E^{(1)}$ with the empty substitution $\sigma^{(1)}$ is the constrained empty clause

$$Q^{(1)} = \leftarrow \ \& \ \texttt{a:Order}, \texttt{b:Product}$$

What we need to check is that $\Sigma' \cup \{$a:Order, b:Product$\}$ is satisfiable. The first unsatisfiability check operates on the initial tableau $S^{(0)} = \Sigma' \cup \{$a:$\neg$Order$\}$. The application of the propagation rule $\rightarrow_{\perp}$ to $S^{(0)}$ produces the tableau $S^{(1)} = \{$a:$\perp\}$. Computation stops here because no other rule can be applied to $S^{(1)}$. Since $S^{(1)}$ is complete and contains a clash, the initial tableau $S^{(0)}$ is unsatisfiable. The second unsatisfiability check operates on the initial tableau $S^{(0)} = \Sigma' \cup \{$b:$\neg$Product$\}$. The only propagation rule applicable to $S^{(0)}$ is $\rightarrow_{\sqsubseteq}$ with respect to the assertion DairyProduct$\sqsubseteq$Product. It produces the tableau $S^{(1)} = \Sigma \cup \{$b:$\neg$Product, b:$\neg$DairyProduct$\sqcup$Product$\}$. By applying $\rightarrow_{\sqcup}$ to $S^{(1)}$ with respect to the concept Product we obtain $S^{(2)} = \Sigma \cup \{$b:$\neg$Product, b:Product$\}$ which presents an evident contradiction. Indeed the application of $\rightarrow_{\perp}$ to $S^{(2)}$ produces the final tableau $S^{(3)} = \{$b:$\perp\}$. Having proved the satisfiability of $\Sigma \cup \{$a:Order, b:DairyProduct$\}$, we have proved the existence of a constrained SLD-refutation for $Q^{(0)}$ in $\mathcal{B}'$. Therefore we can say that $Q_3 \succeq_{\mathcal{B}} Q_9$. It can be easily proved that $Q_9 \not\succeq_{\mathcal{B}} Q_3$.

## 5 Conclusions and Future Work

The hybrid language $\mathcal{AL}$-log was conceived to bridge the gap between Horn clausal logic and description logics in knowledge representation and reasoning. In this paper we have defined a relation of subsumption, called $\mathcal{B}$-subsumption, for constrained Datalog clauses, thus providing a core ingredient for inductive learning in hybrid languages. Indeed $\mathcal{B}$-subsumption can be adopted for structuring the space of inductive hypotheses in concept learning and data mining problems that follow the 'generalization as search' approach and can benefit from the expressive power of $\mathcal{AL}$-log. One such problem is frequent pattern discovery in Mannila's formulation as shown in the illustrative example reported throughout the paper. For the future we plan to investigate theoretical issues such as the learnability of $\mathcal{AL}$-log by paying particular attention to the open world assumption. Also we intend to define ILP techniques for searching $\succeq_{\mathcal{B}}$-ordered spaces of $\mathcal{O}$-queries. This will allow us to carry on the work started in [13] and face new interesting applications, e.g. the Semantic Web.

## References


[1] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2002) 53

[2] Badea, L., Nienhuys-Cheng, S.-W.: A Refinement Operator for Description Logics. In: Cussens, J., Frisch, A.: Inductive Logic Programming. Lecture Notes in Artificial Intelligence, Vol. 1866. Springer-Verlag (2000) 40–59 54

[3] Buntine, W.: Generalized Subsumption and its Application to Induction and Redundancy. Artificial Intelligence **36** (1988) 149–176 59, 61

[4] Ceri, S., Gottlob, G., Tanca, L.: Logic Programming and Databases. Springer (1990) 54, 58

[5] Cohen, W. W., Borgida, A., Hirsh, H.: Computing Least Common Subsumers in Description Logics. In: Swartout, W. R. (ed.): Proc. of the 10th National Conf. on Artificial Intelligence. The AAAI Press / The MIT Press (1992) 754–760 54

[6] Cohen, W. W., Hirsh, H.: Learning the CLASSIC description logic: Thoretical and experimental results. In: Doyle, J., Sandewall, E., Torasso, P. (eds.): Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94). Morgan Kaufmann (1994) 121-133 54

[7] Donini, F. M., Lenzerini, M., Nardi, D., Schaerf, A.: $\mathcal{AL}$-log: Integrating Datalog and Description Logics. Journal of Intelligent Information Systems **10** (1998) 227–252 54, 55, 57, 61

[8] De Raedt, L., Dehaspe, L.: Clausal Discovery. Machine Learning **26** (1997) 99–146 53

[9] Helft, N.: Inductive Generalization: A Logical Framework. In Bratko, I., Lavrač, N. (eds.): Progress in Machine Learning - Proceedings of EWSL87: 2nd European Working Session on Learning. Sigma Press, Wilmslow, U. K. (1987) 149–157 58

[10] Kietz, J.-U., Morik, K.: A Polynomial Approach to the Constructive Induction of Structural Knowledge. Machine Learning **14** (1994) 193–217 54

[11] Levy, A. Y., Rousset, M.-C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence **104** (1998) 165–209 54

[12] Lisi, F. A., Ferilli, S., Fanizzi, N.: Object Identity as Search Bias for Pattern Spaces. In van Harmelen, F. (ed.): ECAI 2002. Proceedings of the 15th European Conference on Artificial Intelligence. IOS Press, Amsterdam (2002) 375–379 59

[13] Lisi, F. A.: An ILP Setting for Object-Relational Data Mining. Ph.D. Thesis, Department of Computer Science, University of Bari, Italy (2002) 63

[14] Mannila, H., Toivonen, H.: Levelwise Search and Borders of Theories in Knowledge Discovery. Data Mining and Knowledge Discovery **1** (1997) 241–258 53

[15] Mitchell, T. M.: Generalization as Search. Artificial Intelligence **18** (1982) 203–226 53

[16] Nienhuys-Cheng, S.-H. and de Wolf, R.: Foundations of Inductive Logic Programming. Lecture Notes in Artificial Intelligence, Vol. 1228. Springer (1997) 53

[17] Reiter, R.: Equality and Domain Closure in First Order Databases. Journal of ACM **27** (1980) 235–249 54

[18] Rouveirol, C., Ventos, V.: Towards Learning in CARIN-$\mathcal{ALN}$. In: Cussens, J., Frisch, A. (eds.): Inductive Logic Programming. Lecture Notes in Artificial Intelligence, Vol. 1866. Springer-Verlag (2000) 191–208 54

[19] Schmidt-Schauss, M., Smolka, G.: Attributive Concept Descriptions with Complements. Artificial Intelligence **48** (1991) 1–26 54

[20] Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: A Logic Framework for the Incremental Inductive Synthesis of Datalog Theories. In Fuchs, N. E. (ed.): Proc. of 7th Int. Workshop on Logic Program Synthesis and Transformation. Lecture Notes in Computer Science Vol. 1463. Springer (1998) 300–321 59

# A Methodology for the Induction of Ontological Knowledge from Semantic Annotations

Nicola Fanizzi, Floriana Esposito, Stefano Ferilli, and Giovanni Semeraro

Dipartimento di Informatica
Università degli Studi di Bari
Campus, Via Orabona 4, 70125 Bari, Italy
{fanizzi,esposito,ferilli,semeraro}@di.uniba.it

**Abstract.** At the meeting point between machine learning and description logics, we investigate on the induction of structural knowledge from metadata. In the proposed methodology, a basic taxonomy of the primitive concepts and roles is preliminarily extracted from the assertions contained in a knowledge base. Then, in order to deal with the inherent algorithmic complexity that affects induction in structured domains, the ontology is constructed incrementally by refining successive versions of the target concept definitions, expressed in richer languages of the Semantic Web, endowed with well-founded reasoning capabilities.

## 1 Introduction

The challenge of the Semantic Web [4] requires the effort for supporting both syntactic and semantic interoperability. The full access to the content of the resources in the Web will be enabled by knowledge bases that are able to maintain not only the mere resources but also information on their meaning (semantic metadata). However, annotating resources after semantic criteria is not a trivial and inexpensive task. Hence, (semi-)automatic tools for the construction, maintenance and processing of this knowledge can be an important factor to boost the realization of the Semantic Web.

Describing resources only in terms of specific data models utilized for their maintenance can be a severe limitation for interoperability at a semantic level. Ontological knowledge is to be employed for organizing and classifying resources on the ground of their meaning. In the proposed frameworks, an ontology is cast as a concept graph built on the ground of a precise lexicon intended for being used by machines. Each class of resources is defined extensionally by the set of the resources it represents, and intensionally by descriptions which possibly account for them and also for instances that may be available in the future. Thus, the problem is how to support the construction of such ontological knowledge.

Logic theories can be employed for representing ontological knowledge. In this context, we present a methodology for the induction of logic definitions for classes of resources and their properties from basic annotations made on instances of

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 65–77, 2003.
© Springer-Verlag Berlin Heidelberg 2003

such resources that may be available in a specific knowledge base. The elicited definitions make up a theory accounting for concepts and relationships that can be a powerful tool for supporting other services, such as reasoning and retrieval.

Devising a learning service that is intended for the Semantic Web, representations that are typical of this context have to be considered. Languages built on top of XML such as *RDF* [20], with vocabularies, such as *Dublin Core* [6], that are generally defined by means of *RDF Schema* [21], have emerged as a standard for the metadata annotation of resources in this context. Besides, the need for further expressiveness and inference capabilities has required the definition of markup languages designed for the Web such as *DAML+OIL* [5] and *OWL* [19], which are envisaged to support reasoning services in the Semantic Web.

Such languages are closely related to the conceptual representations of the *Description Logics* (henceforth DL), which are endowed with well founded semantics and reasoning procedures [1]. In this context, the world state is given by an *A-box* that contains the annotations regarding the resources, while the structural description of their classes and relationships is maintained in a *T-box* or *terminology*.

The induction of structural knowledge like the T-box taxonomies is not new in Machine Learning, especially in the context of *concept formation* [25] where clusters of similar objects are aggregated in hierarchies according to heuristic criteria or similarity measures. Almost all of these methods apply to zero-order representations while, as mentioned above, ontologies are expressed through fragments of first-order logic. Yet, the problem of the induction of structural knowledge in first-order logic (or equivalent representations) turns out to be hard [12].

In the area of *Inductive Logic Programming* (ILP), attempts have been made to extend relational learning techniques toward more expressive languages such as prenex conjunctive normal forms [18] or hybrid representations [9][23][13]. In order to cope with the problem complexity [14], these methods are based on a heuristic search and generally implement bottom-up algorithms [2] that tend to induce overly specific concept definitions which may suffer for poor predictive capabilities. In our methodology, structural knowledge is modeled by means of a representation based on the *object identity* bias [10], which allows us to exploit and evolve our previous work on relational learning by refinement operators [11] for dealing with these different search spaces.

The remainder of the paper is organized as follows. In Section 2 the search space is presented and its properties are discussed. The method for the induction of ontological knowledge is illustrated in Section 3, where its applicability is also briefly analyzed. Possible extensions of the method are discussed in Section 4.

## 2 The Search Space

An ideal data model should be both effective and efficient, meaning that it should be sufficiently expressive for modeling the intended knowledge and also that deductive and inductive inference should be efficiently implementable. In ILP, several solutions have been proposed for the adoption of an expressive fragment

**Table 1.** DL constructors and related interpretation

| **constructor** | **syntax** | **interpretation** $\mathcal{I}$ |
|---|---|---|
| top concept | $\top$ | $\Delta$ |
| bottom concept | $\bot$ | $\emptyset$ |
| concept conjunction | $C_1 \sqcap C_2$ | $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ |
| value restriction | $\forall R.C$ | $\{x \in \Delta \mid \forall y\ (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$ |
| at_most restriction | $\leq n.R$ | $\{x \in \Delta \mid \ \mid \{y \in \Delta \mid (x,y) \in R^{\mathcal{I}}\} \mid \ \leq n\}$ |
| at_least restriction | $\geq n.R$ | $\{x \in \Delta \mid \ \mid \{y \in \Delta \mid (x,y) \in R^{\mathcal{I}}\} \mid \ \geq n\}$ |
| role conjunction | $R_1 \sqcap R_2$ | $R_1^{\mathcal{I}} \cap R_2^{\mathcal{I}}$ |
| inverse role | $R^-$ | $\{(x,y) \in \Delta \times \Delta \mid (y,x) \in R^{\mathcal{I}}\}$ |

of first-order logic endowed with efficient inference procedures. Alternatively, the data model can be expressed by means of DL concept languages for which inference is efficiently computable [7].

Although it can be assumed that annotations and conceptual models are maintained and transported using the XML-based languages mentioned above, the syntax of the core representation adopted here is taken from the standard constructors proposed in the literature [1]. The formalism distinguishes between concepts and roles (resources and properties in RDF) that are described in terms of DL restrictions such as universal and existential quantification (also through upper and lower bounds on cardinalities), role inversion and disjointness.

These DL representations turn out to be both sufficiently expressive and efficient from an inferential viewpoint. Internally, both the assertions (instantiations of concepts or relationships) and the hypotheses (concept definitions) can be represented with the same DL language.

Moreover, they can often be transformed to (constraint) logic programming representations [13] typical of relational learning. However, the supported inference must be adapted for this setting. Indeed, while in the context of DL reasoning the *Open World Assumption* (OWA) is required, in ILP the *Closed World Assumption* (CWA) is generally adopted. Thus, a different notion of explanation becomes essential for testing the candidate solutions of a learning problem.

### 2.1 Knowledge Bases in Description Logics

The theoretical setting of learning in DL spaces requires the definition of syntax and semantics for the proposed representation. Moreover, it is also needed that this specification can be mapped quite straightforwardly to the languages mentioned above.

In a DL language[1], *concepts* $\{C, D, \ldots\}$ are interpreted as subsets of a certain domain of objects and *roles* $\{R, S, \ldots\}$ are interpreted as binary relations. Complex descriptions can be built from atomic concepts ($A$) and primitive roles ($P$) by means of the constructors given in Table 1. In an *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$,

[1] We adopt $\mathcal{ALN}$ augmented with role conjunction and inverse roles.

$\Delta$ is the domain of the interpretation and the functor $\cdot^{\mathcal{I}}$ stands for the interpretation function mapping concepts and roles to their extension.

These constructors are supported by the languages currently developed for the Semantic Web. In DAML+OIL, for instance, the constructor $\forall R.C$ corresponds to the `toclass` restriction, the cardinality restriction $\leq n.R$ corresponds to `maxCardinality`, $R^-$ corresponds to `inverseOf`, etc.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a T-box $\mathcal{T}$ and an A-box $\mathcal{A}$. $\mathcal{T}$ is a set of (acyclic) concept definitions $A \equiv C$, meaning $A^{\mathcal{I}} = C^{\mathcal{I}}$, where $A$ is the concept name and $C$ is a DL description given in terms of the language constructors[2]. $\mathcal{A}$ contains extensional assertions on (primitive) concepts and roles, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$.

The semantic notion of *subsumption* between concepts (or roles) expressed in the given formalism can be given in terms of their interpretations [1]:

**Definition 2.1 (subsumption).** *Given two terms $\alpha$ and $\beta$ in $\mathcal{T}$, $\alpha$* subsumes *$\beta$, denoted by $\alpha \sqsupseteq \beta$, iff $\alpha^{\mathcal{I}} \supseteq \beta^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ of $\mathcal{T}$.*

*Example 2.1.* An example of concept definition in the proposed language:

$$Polygamist \equiv Person \sqcap \forall marriedTo.Person \sqcap \geq 2.marriedTo$$

which translates the sentence *"a Polygamist is a person that is married to at least two other persons"* which is equivalent to the first-order formula:

$$\forall x(Polygamist(x) \leftrightarrow Person(x) \wedge \forall y(marriedTo(x, y) \rightarrow Person(y)) \wedge \\ \exists v, w(marriedTo(x, v) \wedge marriedTo(x, w) \wedge v \neq w)$$

A-box assertions look like:
$Person(john)$, $Person(mary)$, $marriedTo.Person(john, mary)$. Now, if we define

$$Bigamist \equiv Person \sqcap \forall marriedTo.Person \sqcap \geq 2.marriedTo \sqcap \leq 2.marriedTo$$

then it holds that $Polygamist \sqsupseteq Bigamist$

It is possible to note that this DL formalism adopts quite naturally the *unique names assumption* [22] that was extended in the ILP context with the *object identity* bias [11]. This allows us to adapt previous results and methods obtained in the ILP setting to the new learning problem. For instance, object identity gives a way for counting objects, even in the presence of variables, which is important for reasoning with numeric restrictions and also for query answering, e.g. in the context of mining association rules [16].

From a practical point of view, we can consider that the assertions in the A-box can be expressed as RDF annotations. They are to be converted into standard logic programming assertions (facts) at the preliminary stage of concept formation. The induction of candidate concept definitions (*hypotheses*) can be

[2] In the practive, we consider the case of sufficient definitions.

```
<daml:Class rdf:ID="polygamist">
      <rdfs:subClassOf rdf:resource="#person"/>
      <rdfs:subClassOf>
            <daml:Restriction daml:toclass="#person">
                  <daml:onProperty rdf:resource="#marriedTo"/>
            </daml:Restriction>
      </rdfs:subClassOf>
      <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="2">
                  <daml:onProperty rdf:resource="#marriedTo"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>
```

**Fig. 1.** A DAML+OIL translation of the definition for the concept *Polygamist*

performed in a relational learning setting. Conversely, using a suitable wrapper, the output of the inductive algorithm to be presented can be given in one of the ontology languages such as DAML+OIL or OWL, which in turn enforces the knowledge access and reuse in the Semantic Web perspective. For instance, translating the concept definition of the previous example into DAML+OIL, we have the description in Figure 1.

In inductive reasoning, it is often necessary to test the coverage of induced candidate hypotheses (for a T-box) with respect to the examples (A-box assertions). Coverage also determines the decisions on the possible refinement of such hypotheses. However, applying these methods in a DL setting, problems arise with the OWA which conflicts with the CWA that is commonly adopted in the context of learning or databases. A possible solution, as discussed in [3], is that of considering an *epistemic operator* $K$, such as the one proposed in [8].

For instance, to test whether a simple definition $A \equiv D \sqcap \forall R.C$ covers $A(e)$, the universal restriction $\forall R.C$ cannot verified under OWA, unless $\forall R.C(e)$ is explicitly asserted in the knowledge base $\mathcal{K}$. Conversely $\forall KR.C$ is verified if all the *known* $R$-fillers $R(e, o_1)$, $R(e, o_2), \ldots$ in $\mathcal{A}$ also verify $C$:

**Definition 2.2 (coverage).** *Given the knowledge base $\langle \mathcal{T}, \mathcal{A} \rangle$, the sufficient definition of a concept $A \equiv C$* covers *an assertion (i.e. an example) $A(e)$ iff*

$$cl(\mathcal{T} \cup \{C\}, \mathcal{A}) \models A(e)$$

The closure operator $cl$ is the *role-closure* of the knowledge base, i.e. $KR$ is considered instead of $R$. OWA in a similar ILP setting is discussed in [3].

The learning problem, for an unsupervised learning case, can be now formally defined as follows (adapted from [3]):

**Definition 2.3 (learning problem).** *Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, for each concept $C$ with assertions in $\mathcal{A}$, supposed that $\mathcal{T} \not\models \mathcal{A}$, induce a set of concept definitions (hypotheses) $\mathcal{T}_C = \{C_1 \equiv D_1,\ C_2 \equiv D_2,\ \ldots\}$ such that $\mathcal{T} \cup \mathcal{T}_C \models \mathcal{A}$ that is $\forall A(e) \in \mathcal{A}$: $\mathcal{T} \cup \mathcal{T}_C$ covers $A(e)$*

Thus, the problem requires to find definitions $\mathcal{T}_C$ for undefined concepts for (new) assertions in the A-box. $\mathcal{T}$ is to be regarded as a sort of background knowledge (possibly imported from higher level ontologies), which can be supposed to be correct but also incapable to explain the assertions in the A-box.

### 2.2 Ordering the Search Space

The induction of the definitions in $\mathcal{T}_C$ can be modeled as a search once an order is imposed on the conceptual representations. The properties of the *search space* depend on the subsumption order that is adopted. This notion induces a generalization model (a quasi-order) that gives a criterion for traversing the space of solutions by means of suitable operators (see [17] for a review of the related orders on clausal spaces).

In an ILP context, this notion can be compared to *OI-implication*, an object identity compliant form of implication [11], which is particularly suitable for structured representations, such as T-boxes. Moreover, refinement operators for clausal search spaces whose algebraic structure is determined by the order induced by OI-implication have also been proposed [10].

Within an ordered space, refinement operators represent the theoretical key that allows the treatment of learning as decoupled into search and heuristics. They have been widely investigated for refining logic programs in ILP [17][10]. Refinement operators exploit the notion of order imposed on the representation, that makes the set of possible concept definitions a search space.

In our case, we exploit the order induced by the notion of subsumption given in Definition 2.1:

**Definition 2.4 (DL refinement operators).** *Given a search space* $(\mathcal{S}, \sqsupseteq)$ *of hypotheses for a concept, a* downward *(respectively* upward*)* refinement operator $\rho$ *(resp.* $\delta$*) is a mapping from* $\mathcal{S}$ *to* $2^{\mathcal{S}}$*, such that* $H' \in \rho(H)$ *implies* $H \sqsupseteq H'$ *(resp.* $H' \in \delta(H)$ *implies* $H' \sqsupseteq H$*).*

## 3 Induction of T-Boxes

The methodology for concept formation proposed in this work (and also the terminology) is inspired from the algorithm given in [14]. However, we have applied refinement operators and heuristics that are typically employed in ILP. The algorithm applied in our method is presented in Figure 2.

Initially, the basic taxonomy of primitive concepts and roles is built from the knowledge available in the starting A-box $\mathcal{A}$ (if also the T-box $\mathcal{T}$ is non-empty, the A-box is augmented with assertions obtained by saturation with respect to the T-box). This step singles out domains and ranges of the roles, the underlying subsumption relationships between the concept extensions (making up a hierarchy or a graph) and also all the pairwise disjointness relationships that can directly or indirectly be detected. This is exploited to induce the candidate clusters of concepts. Indeed, mutually disjoint concepts will require non-overlapping

```
induce_T-box(A,T,T')
input    A: A-box; T: T-box
output   T':T-box

if T ≠ ∅ then  A := A ∪ saturate(A,T)
initialize_basic_taxonomy(A, T,MDCs)
repeat
        MDC := select(MDCs)
        for each concept C in MDC
                MSG := define(C,A,δ)
                q := evaluate(MSG,MDC)
                while q < ε_q do
                        MSG := refine(MSG,MDC,ρ)
                        q := evaluate(MSG,MDC)
        MSGs := MSGs ∪ MSG
        MDCs := MDCs \ MDC
until MDCs is empty
for each MSG in MSGs
        MGD := generalize(MSG,MDC,δ)
        store(MGD,MGDs)
T' := MGDs ∪ T
return T'
```

**Fig. 2.** An algorithm for T-box induction based on refinement operators

definitions. A $MDC$ stands here for the *maximal set of mutually disjoint concepts*, i.e. a cluster of disjoint subconcepts of the same concept.

All the non-primitive subconcepts within a $MDC$ need a discriminating definition, that is to be induced as a result of a separate supervised learning task. Thus, a loop is repeated looking for a *maximally specific generalization* ($MSG$) for each concept in the selected $MDC$. This can be achieved using an upward operator $\delta$ by which it is possible to search for more and more general definition incrementally. Moreover, the A-box $\mathcal{A}$ (that is the extensions of primitive concepts and roles) is exploited in heuristics that may focus this search.

When a $MSG$ turns out to be too poor at defining a concept in the context of a $MDC$, i.e. it covers negative examples represented by the instances of concepts that must be disjoint, it has to be refined by means of a specializing downward operator. A given threshold $\epsilon_q$ states the minimum quality for a candidate $MSG$ of a subconcept. The heuristics that can be employed in the evaluation of a candidate $MSG$ and the refinement operator will be specified later on.

In this setting, it is assumed that the language bias is adequate for the induction of the target definitions. However, sometimes the heuristic search may fail, denouncing the inadequacy of the vocabulary in use. In such cases, new concepts or roles may be introduced through a process that is similar to constructive

induction, which can be adapted from ILP [24], for building discriminating definitions.

Finally, all the $MSG$s need to undergo an upward refinement step (generalization) in order to avoid overfitting and increase the predictiveness of the definition towards new unseen instances. The result is a *most general discrimination* ($MGD$) to be induced for each concept.

To make the problems more intelligible, the discussion on the heuristics is separated from the one on the refinement operators. Practical implementations of the method will have to integrate the two issues.

### 3.1 Heuristics for Hypothesis Evaluation

The evaluation of a concept definition which is candidate to become a $MSG$ should take into account the coverage of positive and negative examples in the learning problem. This problem concerns the induction of a general definition of a concept from positive examples and against the negative examples represented by the instances of the other concepts within the same $MDC$.

Due to the breadth of the search space, heuristics should be used together with the underlying generalization model in order to focus the search. Intuitively, a good hypothesis should cover as many positive examples as possible and reject the negative ones. Moreover, typically other limitations are made upon the size of the hypotheses, in favor of the simpler ones (those containing less restrictions).

In the algorithm presented, a possible form for the evaluation function of the $i$-th concept definition with respect to the $j$-th $MDC$ may be:

$$evaluate(MSG_i, MDC_j) = w_p \cdot pos_{ij} - w_n \cdot neg_{ij} - w_s \cdot size_i$$

here $pos_{ij}$ and $neg_{ij}$ are determined by the rate of examples covered by the candidate hypothesis over the examples in the $MDC$, while $size_i$ should be calculated on the ground of its syntactic complexity (each term adds 1 to the size plus the size of the nested concept description, if any). Although this may seem quite naive, it has proven effective in ILP context and it is also efficient even depending on the coverage procedure. It is worthwhile to recall that the efficiency of refinement also depends on the choice of refinement operators and their properties. Other heuristics can be found in [15].

Once the incorrect concept definition is found, the available examples can help decide which refinement rule to apply e.g. on literals to be added or removed, etc. Besides, following [14], it is possible to detect the roles (and their inverse) that allow the coverage of examples of disjoint concepts so that they become candidates for a further localized refinement. Similarly, it is possible to blame restrictions that need an upward refinement. Rather than keeping separate the generation and test of the hypotheses, increasing the system efficiency should be pursued by coupling heuristics to the refinement operators.

### 3.2 Refinement Operators for T-boxes

Given the ordering relationship defined for the space of hypotheses, it is possible to specify how to traverse this search space by means of refinement operators.

Several different properties of the refinement operators have been investigated. Among them the most important are *completeness*, *ideality*, *minimality* and *non-redundancy*. Recently, there have been some attempts to transpose previous work on clausal logics to this specific learning task in a DL context [3]. The definition of a complete downward refinement operator for the $\mathcal{ALER}$ description logic is given. Here it suffices to define a simpler operator, due to the different concept language adopted.

It is supposed to consider a DL-language containing *literals* (atomic concepts or restrictions on roles) that can be added or dropped to the concept definitions, in order to, respectively, specialize or generalize them.

**Definition 3.1 (downward refinement operator).** *A downward refinement operator $\rho$ is defined:*

- $D \sqcap L \in \rho(D)$ *if* $L$ *is a literal*
- $D \sqcap \forall R.C_1 \in \rho(D \sqcap \forall R.C_2)$ *with* $C_1 \sqsubseteq C_2$ *or* $C_1 = \bot$
- $D \sqcap \forall R_1.C \in \rho(D \sqcap \forall R_2.C)$ *with* $R_2 \sqsubseteq R_1$
- $D \sqcap \leq n.R \in \rho(D \sqcap \leq m.R)$ *if* $n < m$
- $D \sqcap \geq n.R \in \rho(D \sqcap \geq m.R)$ *if* $n > m$

Role descriptions $R$ can be specialized by adding primitive roles: $R \sqcap P$.

*Example 3.1.* Suppose that we have a language with the concept atoms $A_i, i = 1, \ldots, 4$, and primitive roles $R_1$ and $R_2$, a possible refinement chain, starting from the top, is given by:
$A_1 \in \rho(\top)$
$A_1 \sqcap \forall(R_1 \sqcap R_2).A_2 \in \rho(A_1)$
$A_1 \sqcap \forall(R_1 \sqcap R_2).A_2 \sqcap \leq n.R_2 \in \rho(A_1 \sqcap \forall(R_1 \sqcap R_2).A_2)$
$A_1 \sqcap \forall R_1.A_2 \sqcap \leq n.R_2 \in \rho(A_1 \sqcap \forall(R_1 \sqcap R_2).A_2 \sqcap \leq n.R_2)$ and so on.

It is now straightforward to define a dual upward (i.e. generalizing) operator $\delta$ that searches for other hypotheses by generalizing or dropping restrictions.

**Definition 3.2 (upward refinement operator).** *An upward refinement operator $\delta$ is defined:*

- $D \in \delta(D \sqcap L)$ *if* $L$ *is an literal*
- $D \sqcap \forall R.C_1 \in \delta(D \sqcap \forall R.C_2)$ *with* $C_1 \sqsupseteq C_2$ *or* $C_1 = \top$
- $D \sqcap \forall R_1.C \in \delta(D \sqcap \forall R_2.C)$ *with* $R_1 \sqsupseteq R_2$
- $D \sqcap \leq n.R \in \delta(D \sqcap \leq m.R)$ *if* $n > m$
- $D \sqcap \geq n.R \in \delta(D \sqcap \geq m.R)$ *if* $n < m$

Role descriptions $R \sqcap P$ can be generalized by dropping primitive roles: $R$.

Defining both operators upward and downward allows for the extension of the method toward a full incremental algorithm for knowledge revision working on dynamically changing A-boxes, where each new assertion is processed singularly.

### 3.3 Discussion

Generally in a DL search space there are many forms of characterization of a concept obtainable when looking for a $MSG$. Yet it can be shown that the difference is merely syntactic. All these forms can be reduced to a single one through a process of normalization (see [14] for the normalization rules in the adopted logic). For example, conjunctions of $n$ restrictions $\forall R.C_i$ on the same role $R$ ($i = 1, \ldots, n$) can be grouped into $\forall R.(C_1 \sqcap \cdots \sqcap C_n)$.

The choice of the formalism guarantees the efficiency of the coverage test (and subsumption in general) [7], unless, as suggested in the previous section, the algorithm is extended by allowing new terms to be introduced during the downward refinement steps.

As regards the algorithm, the initial characterization of a concept is modeled in our method like an upward search for generalizations. This is similar to the approach based on the calculation of the *least common subsumer* [2]. In [3] a different approach is proposed. The algorithm should start from the most general definition $\top$, and then it would repeatedly apply a downward refinement operator $\rho$ up to finding discriminating generalizations for the target concepts.

The original method described in [14] induces the final definitions by means of incomplete specialization and generalization operators. They are not guaranteed to find a definition, since they limit the search in order to be more efficient. For example, let us consider the case of the generalization in the induction of a $MSG$ in the original algorithm. It follows a predefined schema that requires first to drop restrictions, as long as instances of disjoint concepts are not covered, and then to generalize the remaining ones. This forces an order in the refinement graph which may not lead to the correct definitions. Completeness, like the other properties of the refinement operators depending on the search space is an indispensable property. It can be shown that the operators proposed in this paper are indeed complete. However, they might turn out to be very redundant and non-minimal.

In order to avoid these problems, the refinement operators should be redefined, by imposing a sort of order on the possible refinements and decomposing the refinement steps as much as possible into fine-grained ones. This is often very difficult and prone to collide with the completeness of the operators. The properties of the refinement operators to be required are to be decided when further knowledge is available concerning the search space. Properness and completeness are indicated for spaces with dense solutions, whereas non-redundancy applies better with spaces with rare solutions.

Although the overall algorithm adopts a batch strategy, the intrinsic incrementality of induction regarded as search, offers the possibility of an extension of the method toward revising or restructuring the existing T-boxes in the presence of enlarged A-boxes due to the availability of new assertions.

## 4 Conclusions and Future Work

Structural knowledge is expected to play an important role in the Semantic Web. Yet, while techniques for deductive reasoning and querying for such knowledge bases are now very well assessed, their construction is a hard task for knowledge engineers even for limited domains; they could be supported by (semi-)automatic inductive tools that still required a lot of investigation. We have presented a method for building structural knowledge in description logics.

The feasibility of the method is related to the transposition in a different representation language of techniques developed in the area of ILP. The mere application of the existing pure ILP systems to learning structural knowledge has demonstrated its limits (e.g. see [14]) both because of the change of the representation and for the properties of the inductive algorithms and of the heuristics devised to cope with the combinatorial complexity of the learning problem. A deeper investigation on the properties of the refinement operators on DL languages is required. In order to increase the efficiency of learning, redundancies during the search for solutions are to be avoided. This can be done by defining minimal refinement operators [3].

The method illustrated in this paper is currently being implemented in a system (CSKA, *Clustering of Structural Knowledge from an A-box*) that is expected to induce T-boxes in a conceptual language such as DAML+OIL from A-boxes made up of RDF annotations. Representing the assertions and concept definitions by means of these languages will allow to design a learning service for the Semantic Web.

The proposed framework could be extended along three directions. First, a more expressive language bias could be chosen, allowing the possibility of disjunctive definition. Besides, the transitivity of relations would allow to learn recursive concepts. Secondly, an incremental setting based on searching DL spaces through refinement operators could be exploited for tasks of diagnosis and revision of existing knowledge bases, as long as new information becomes available. ILP is a learning paradigm that is able to deal quite naturally with explicit background knowledge that may be available. Then, a promising direction seems also to investigate hybrid representations, where clausal logic descriptions are mixed with description logics, the latter accounting for the available ontological knowledge.

## References

[1] F. Baader, D. Calvanese, D. McGuinnes, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003. 66, 67, 68

[2] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 96–101. Morgan Kaufmann, 1999. 66, 74

[3] L. Badea and S.-H. Nienhuys-Cheng. A refinement operator for description logics. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *LNAI*, pages 40–59. Springer, 2000. 69, 73, 74, 75

[4] T. Berners-Lee. Semantic Web road map. Technical report, W3C, 1998. http://www.w3.org/DesignIssues/Semantic.html. 65

[5] DAML+OIL. DAML+OIL ontology markup language reference, 2001. http://www.daml.org/2001/03/reference. 66

[6] DC. Dublin Core language reference, 2003. http://www.purl.org/dc. 66

[7] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 458–463, 1991. 67, 74

[8] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274, 1998. 69

[9] F. Donini, M. Lenzerini, D. Nardi, and M. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and description logics. *Journal of Intelligent Information Systems*, 10:227–252, 1998. 66

[10] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. A generalization model based on OI-implication for ideal theory refinement. *Fundamenta Informaticæ*, 47:15–33, 2001. 66, 70

[11] F. Esposito, N. Fanizzi, S. Ferilli, and G. Semeraro. OI-implication: Soundness and refutation completeness. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 847–852, Seattle, WA, 2001. 66, 68, 70

[12] D. Haussler. Learning conjuntive concepts in structural domains. *Machine Learning*, 4:7–40, 1989. 66

[13] J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Proceedings of the 12th International Conference on Inductive Logic Programming*, volume 2583 of *LNAI*, pages 117–132, Sydney, 2002. Springer. 66, 67

[14] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(2):193–218, 1994. 66, 70, 72, 74, 75

[15] N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In S. Džeroski and P. Flach, editors, *Proceedings of the 9th International Workshop on Inductive Logic Programming*, volume 1634 of *LNAI*, pages 174–185. Springer, 1999. 72

[16] F. Lisi, N. Fanizzi, and S. Ferilli. Object identity as search bias for pattern spaces. In F. V. Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 375–379, Lyon, 2002. IOS Press. 68

[17] S. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *LNAI*. Springer, 1997. 70

[18] S. Nienhuys-Cheng, W. V. Laer, J. Ramon, and L. D. Raedt. Generalizing refinement operators to learn prenex conjunctive normal forms. In *Proceedings of the International Conference on Inductive Logic Programming*, volume 1631 of *LNAI*, pages 245–256. Springer, 1999. 66

[19] OWL. Web Ontology Language Reference Version 1.0. Technical report, W3C, 2003. http://www.w3.org/TR/owl-ref. 66

[20] RDF. RDF Model and Syntax Specification. Technical report, W3C, 1999. http://www.w3.org/TR/REC-rdf-syntax. 66

[21] RDF-Schema. RDF Vocabulary Description Language 1.0: RDF Schema. Technical report, W3C, 2003. http://www.w3c.org/TR/rdf-schema. 66

[22] R. Reiter. Equality and domain closure in first order databases. *Journal of ACM*, 27:235–249, 1980. 68

[23] C. Rouveirol and V. Ventos. Towards learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *LNAI*, pages 191–208. Springer, 2000. 66

[24] I. Stahl. Predicate invention in Inductive Logic Programming. In L. D. Raedt, editor, *Advances in Inductive Logic Programming*, pages 34–47. IOS Press, 1996. 72

[25] K. Thompson and P. Langley. Concept formation in structured domains. In D. Fisher, M. Pazzani, and P. Langley, editors, *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, 1991. 66

# Qualitative Spatial Reasoning in a Logical Framework

Alessandra Raffaetà[1], Chiara Renso[2], and Franco Turini[3]

[1] Dipartimento di Informatica – Università Ca' Foscari Venezia
raffaeta@dsi.unive.it
[2] ISTI CNR – Pisa
c.renso@isti.cnr.it
[3] Dipartimento di Informatica – Università di Pisa
turini@di.unipi.it

**Abstract.** In this paper we present an approach to qualitative spatial reasoning based on the spatio-temporal language STACLP [18]. In particular, we show how the topological 9-intersection model [7] and the direction relations based on projections [16] can be modelled in such a framework. STACLP is a constraint logic programming language where formulae can be annotated with labels (annotations) and where relations between these labels can be expressed by using constraints. Annotations are used to represent both time and space.

## 1 Introduction

One of the most promising directions of current research in Geographical Information Systems (GISs) focuses on the development of reasoning formalisms that merge contributions from both Artificial Intelligence (AI) and mathematical research areas in order to express *spatial qualitative reasoning* [5, 1]. Such field addresses several aspects of space including topology, orientation, shape, size, and distance.

Qualitative Spatial Reasoning has been recognized as a major point in the future developments of GIS [9]. This can be seen as part of the wider problem of designing high-level GIS user interfaces (see e.g., [9, 8, 2]), which has been addressed from different points of view. Today GIS technology is capable of efficiently storing terabytes of data, but it lacks support for intuitive or common sense reasoning on such data. In this context, the key point is to abstract away from the huge amount of numerical data and to define formalisms that allow the user to specify *qualitative* queries.

Most qualitative approaches focus on the description of the relationships between spatial entities. Prominent examples are *direction* and *topological relations*, i.e., spatial relations that are invariant under topological transformations like translation, rotation, scaling. Many approaches to spatial topological relations can be found in the literature coming from both mathematics and philosophical logics: the *RCC (Region Connection Calculus)* [19], originating from a proposal

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 78–90, 2003.
© Springer-Verlag Berlin Heidelberg 2003

of Clarke [3], the *9-intersection model* proposed by Egenhofer [7, 15] and the *CBM (Calculus-based Method)* [4], adopted by the OpenGIS standard [6].

Direction relations deal with cardinal points such as *north, south-west*. In the *projection-based* approaches [20, 16] the space is divided using horizontal and vertical lines passing through the reference point or delimiting the reference object. The *cone-based* approaches [17, 10], rely on the idea of partitioning the space around a reference object into four (or eight) partitions of 90 or 45 degrees. Finally a recent approach [12, 13] allows the representation of cardinal directions between objects by using their exact geometries.

In this paper we propose an approach to topological and direction relations based on a constraint logic programming language enriched with annotations, called *STACLP* [18]. Such a language extends Temporal Annotated Constraint Logic Programming (TACLP) [11], a constraint logic programming language with temporal annotations, by adding spatial annotations. The pieces of spatio-temporal information are given by pairs of annotations which specify the spatial extent of an object at a certain time period. The use of annotations makes time and space explicit but avoids the proliferation of spatial and temporal variables and quantifiers. Moreover, it supports both definite and indefinite spatial and temporal information, and it allows one to establish a dependency between space and time, thus permitting to model continuously moving points and regions.

In [18] this language is used to perform quantitative spatio-temporal reasoning on geographical data. The present paper, being focussed on the representation of topological and direction relations, concentrates, instead, on the qualitative reasoning capabilities of the framework.

*Overview of the paper.* In § 2, we introduce the language STACLP, and in § 3 we define its semantics using a logical meta-interpreter. In § 4 we show how topological and direction relations can be modelled in STACLP. In § 5 we give an example aimed at illustrating the expressiveness of our approach. Finally, in § 6, we conclude with a discussion and future work.

## 2 STACLP: A Spatio-temporal Language

We introduce an extension to TACLP where both temporal and spatial information can be dealt with. The resulting framework is called Spatio-Temporal Annotated Constraint Logic Programming (STACLP) [18].

### 2.1 Time and Space

Time can be discrete or dense. Time points are totally ordered by the relation $\leq$. We denote by $\mathcal{T}$ the set of time points and we suppose to have a set of operations (e.g., the binary operations $+$, $-$) to manage such points. The time-line is left-bounded by 0 and open to the future, with the symbol $\infty$ used to denote a time point that is later than any other. A *time period* is an interval $[r, s]$ with $r, s \in \mathcal{T}$ and $0 \leq r \leq s \leq \infty$, which represents the convex, non-empty set of time points $\{t \mid r \leq t \leq s\}$. Thus the interval $[0, \infty]$ denotes the whole time line.

Analogously space can be discrete or dense and we consider as spatial regions *rectangles* represented as $[(x_1, x_2), (y_1, y_2)]$, where $(x_1, y_1)$ and $(x_2, y_2)$ denote the lower-left and upper-right vertex of the rectangle, respectively. More precisely, $[(x_1, x_2), (y_1, y_2)]$ models the region $\{(x, y) \mid x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\}$. Rectangles are the two-dimensional counterpart of convex sets of time points.

### 2.2 Annotations and Annotated Formulae

An *annotated formula* is of the form $A\,\alpha$ where $A$ is an atomic formula and $\alpha$ an annotation. We define three kinds of temporal and spatial annotations inspired by similar principles:

- $\mathtt{at}\,T$ and $\mathtt{atp}\,(X, Y)$ are used to express that a formula holds in a time or spatial point.
- $\mathtt{th}\,I$, $\mathtt{thr}\,R$ are used to express that a formula holds *throughout*, i.e., at *every* point, in the temporal interval or the spatial region, respectively.
- $\mathtt{in}\,I$, $\mathtt{inr}\,R$ are used to express that a formula holds at *some* point(s), in the interval or the region, respectively. They account for indefinite information.

The set of annotations is endowed with a partial order relation $\sqsubseteq$. Given two annotations $\alpha$ and $\beta$, the intuition is that $\alpha \sqsubseteq \beta$ if $\alpha$ is "less informative" than $\beta$ in the sense that for all formulae $A$, $A\,\beta \Rightarrow A\ \ \alpha$. This partial order is used in the definition of new inference rules. In addition to Modus Ponens, STACLP has the two inference rules below:

$$\frac{A\,\alpha \qquad \gamma \sqsubseteq \alpha}{A\,\gamma}\ \ \textit{rule}\ (\sqsubseteq) \qquad\qquad \frac{A\,\alpha \qquad A\,\beta \qquad \gamma = \alpha \sqcup \beta}{A\,\gamma}\ \ \textit{rule}\ (\sqcup)$$

The rule $(\sqsubseteq)$ states that if a formula holds with some annotation, then it also holds with all annotations that are smaller according to the lattice ordering. The rule $(\sqcup)$ says that if a formula holds with some annotation $\alpha$ and the same formula holds with another annotation $\beta$ then it holds with the least upper bound $\alpha \sqcup \beta$ of the two annotations.

Next, we introduce the *constraint theory for temporal and spatial annotations*. A constraint theory is a non-empty, consistent first order theory that axiomatizes the meaning of the constraints. Besides an axiomatization of the total order relation $\leq$ on the set of points, the constraint theory includes the axioms in Table 1 defining the partial order on temporal and spatial annotations. The first two axioms state that $\mathtt{th}\,I$ and $\mathtt{in}\,I$ are equivalent to $\mathtt{at}\,t$ when the time period $I$ consists of a single time point $t$. Next, if a formula holds at every element of a time period, then it holds at every element in all sub-periods of that period ($(\mathtt{th}\ \sqsubseteq)$ axiom). On the other hand, if a formula holds at some points of a time period then it holds at some points in all periods that include this period ($(\mathtt{in}\ \sqsubseteq)$ axiom). The axioms for spatial annotations are analogously defined.

**Table 1.** Axioms for the partial order on annotations

| | |
|---|---|
| (at th) | $\mathtt{at}\, t = \mathtt{th}\,[t,t]$ |
| (at in) | $\mathtt{at}\, t = \mathtt{in}\,[t,t]$ |
| (th $\sqsubseteq$) | $\mathtt{th}\,[s_1,s_2] \sqsubseteq \mathtt{th}\,[r_1,r_2] \Leftrightarrow r_1 \leq s_1,\ s_2 \leq r_2$ |
| (in $\sqsubseteq$) | $\mathtt{in}\,[r_1,r_2] \sqsubseteq \mathtt{in}\,[s_1,s_2] \Leftrightarrow r_1 \leq s_1,\ s_2 \leq r_2$ |
| (atp thr) | $\mathtt{atp}\,(x,y) = \mathtt{thr}\,[(x,x),(y,y)]$ |
| (atp inr) | $\mathtt{atp}\,(x,y) = \mathtt{inr}\,[(x,x),(y,y)]$ |
| (thr $\sqsubseteq$) | $\mathtt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqsubseteq \mathtt{thr}\,[(x'_1,x'_2),(y'_1,y'_2)] \Leftrightarrow$ $x'_1 \leq x_1,\ x_2 \leq x'_2, y'_1 \leq y_1,\ y_2 \leq y'_2$ |
| (inr $\sqsubseteq$) | $\mathtt{inr}\,[(x'_1,x'_2),(y'_1,y'_2)] \sqsubseteq \mathtt{inr}\,[(x_1,x_2),(y_1,y_2)] \Leftrightarrow$ $x'_1 \leq x_1,\ x_2 \leq x'_2, y'_1 \leq y_1,\ y_2 \leq y'_2$ |

### 2.3 Combining Spatial and Temporal Annotations

In order to obtain spatio-temporal annotations the spatial and temporal annotations are combined by considering pairs of annotations as a new class of annotations. Let us first introduce the general idea of pairing of annotations.

**Definition 1.** *Let* $(A, \sqsubseteq_A)$ *and* $(B, \sqsubseteq_B)$ *be two disjoint classes of annotations with their partial order. Their* pairing *is the class of annotations* $(A * B, \sqsubseteq_{A*B})$ *defined as* $A * B = \{\alpha\beta, \beta\alpha \mid \alpha \in A,\ \beta \in B\}$ *and* $\gamma_1 \sqsubseteq_{A*B} \gamma_2$ *whenever*

$$((\gamma_1 = \alpha_1\beta_1 \wedge \gamma_2 = \alpha_2\beta_2) \vee (\gamma_1 = \beta_1\alpha_1 \wedge \gamma_2 = \beta_2\alpha_2)) \ \wedge (\alpha_1 \sqsubseteq_A \alpha_2 \ \wedge \ \beta_1 \sqsubseteq_B \beta_2)$$

In our case the spatio-temporal annotations are obtained by considering the pairing of spatial and temporal annotations.

**Definition 2 (Spatio-temporal Annotations).** *The class of spatio-temporal annotations is the pairing of the spatial annotations* Spat *built from* `atp`, `thr` *and* `inr` *and of the temporal annotations* Temp, *built from* `at`, `th` *and* `in`, *i.e.* Spat∗Temp.

To clarify the meaning of our spatio-temporal annotations, we present some examples of their formal definition in terms of `at` and `atp`. Let $t$ be a time point, $J = [t_1, t_2]$ be a time period, $s = (x, y)$ be a spatial point and $R = [(x_1, x_2), (y_1, y_2)]$ be a rectangle.

⋆ The equivalent annotated formulae $A\,\mathtt{atp}\,s\,\mathtt{at}\,t$ and $A\,\mathtt{at}\,t\,\mathtt{atp}\,s$ mean that $A$ holds at time point $t$ in the spatial point $s$.

⋆ The annotated formula $A\,\mathtt{thr}\,R\,\mathtt{th}\,J$ means that $A$ holds *throughout* the time period $J$ and at every spatial point in $R$. The definition of such a formula in terms of `atp` and `at` is:

$$A\ \mathtt{thr}\,R\,\mathtt{th}\,J \ \Leftrightarrow\ \forall t \in J.\ \forall s \in R.\ A\ \mathtt{atp}\,s\,\mathtt{at}\,t.$$

The formula $A\,\mathtt{th}\,J\,\mathtt{thr}\,R$ is equivalent to the formula above because one can be obtained from the other just by swapping the two universal quantifiers.

**Table 2.** Axioms for least upper bound of annotations

| | |
|---|---|
| (1) | $\texttt{thr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[t_1,t_2] \sqcup \texttt{thr}\,[(x_1,x_2),(z_1,z_2)]\texttt{th}\,[t_1,t_2] = \texttt{thr}\,[(x_1,x_2),(y_1,z_2)]\texttt{th}\,[t_1,t_2] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$ |
| (1′) | axiom obtained by swapping the annotations in (1). |
| (2) | $\texttt{thr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[t_1,t_2] \sqcup \texttt{thr}\,[(z_1,z_2),(y_1,y_2)]\texttt{th}\,[t_1,t_2] = \texttt{thr}\,[(x_1,z_2),(y_1,y_2)]\texttt{th}\,[t_1,t_2] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$ |
| (2′) | axiom obtained by swapping the annotations in (2). |
| (3) | $\texttt{thr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[s_1,s_2] \sqcup \texttt{thr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[r_1,r_2] = \texttt{thr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$ |
| (3′) | axiom obtained by swapping the annotations in (3). |
| (4) | $\texttt{inr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[s_1,s_2] \sqcup \texttt{inr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[r_1,r_2] = \texttt{inr}\,[(x_1,x_2),(y_1,y_2)]\texttt{th}\,[s_1,r_2] \Leftrightarrow s_1 \leq r_1, r_1 \leq s_2, s_2 \leq r_2$ |
| (5) | $\texttt{in}\,[t_1,t_2]\texttt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \texttt{in}\,[t_1,t_2]\texttt{thr}\,[(x_1,x_2),(z_1,z_2)] = \texttt{in}\,[t_1,t_2]\texttt{thr}\,[(x_1,x_2),(y_1,z_2)] \Leftrightarrow y_1 \leq z_1, z_1 \leq y_2, y_2 \leq z_2$ |
| (6) | $\texttt{in}\,[t_1,t_2]\texttt{thr}\,[(x_1,x_2),(y_1,y_2)] \sqcup \texttt{in}\,[t_1,t_2]\texttt{thr}\,[(z_1,z_2),(y_1,y_2)] = \texttt{in}\,[t_1,t_2]\texttt{thr}\,[(x_1,z_2),(y_1,y_2)] \Leftrightarrow x_1 \leq z_1, z_1 \leq x_2, x_2 \leq z_2$ |

⋆ The annotated formula $A\,\texttt{thr}\,R\,\texttt{in}\,J$ means that there exist(s) *some* time point(s) in the time period $J$ in which $A$ holds throughout the region $R$. The definition of such a formula in terms of `atp` and `at` is:

$$A\ \texttt{thr}\,R\,\texttt{in}\,J \ \Leftrightarrow\ \exists t \in J.\ \forall s \in R.\ A\ \texttt{atp}\,s\,\texttt{at}\,t.$$

In this case swapping the annotations swaps the universal and existential quantifiers and hence results into a different annotated formula $A\,\texttt{in}\,J\,\texttt{thr}\,R$, meaning that for every spatial point in the region $R$, $A$ holds at some time point(s) in $J$. Thus we can state *snow* $\texttt{thr}\,R\,\texttt{in}\,[jan, mar]$ in order to express that there exists a time period between January and March in which the whole region $R$ is completely covered by the snow. On the other hand *snow* $\texttt{in}\,[jan, mar]\,\texttt{thr}\,R$ expresses that from January to March each point of the region $R$ will be covered by the snow, but different points can be covered in different time instants.

### 2.4 Least Upper Bound and Its Constraint Theory

For technical reasons related to the properties of annotations (see [11, 18]), we restrict the rule (⊔) to least upper bounds that produce valid, new annotations, i.e., rectangular regions and temporal components which are time periods. Thus we consider the least upper bound in the cases illustrated in Table 2.

Axioms (1), (1′), (2) and (2′) allow one to enlarge the region in which a property holds in a certain interval. If a property $A$ holds both throughout a region $R_1$ and throughout a region $R_2$ in every point of the time period $I$ then it holds throughout the region which is the union of $R_1$ and $R_2$, throughout $I$. Notice that the constraints on the spatial variables ensure that the resulting region is

still a rectangle. Axioms (3) and (3′) concern the temporal dimension: if a property $A$ holds throughout a region $R$ and in every point of the time periods $I_1$ and $I_2$ then $A$ holds throughout the region $R$ in the time period which is the union of $I_1$ and $I_2$, provided that $I_1$ and $I_2$ are overlapping. By using axiom (4) we can prove that if a property $A$ holds in some point(s) of region $R$ throughout the time periods $I_1$ and $I_2$ then $A$ holds in some point(s) of region $R$ throughout the union of $I_1$ and $I_2$, provided that such intervals are overlapping. Finally, the last two axioms allow to enlarge the region $R$ in which a property holds in the presence of an `in` temporal annotation.

### 2.5 Clauses

The clausal fragment of STACLP, which can be used as an efficient spatio-temporal programming language, consists of clauses of the following form:

$$A\,\alpha\beta \leftarrow C_1, \ldots, C_n, B_1\,\alpha_1\beta_1, \ldots, B_m\,\alpha_m\beta_m \quad (n, m \geq 0)$$

where $A$ is an atom, $\alpha$, $\alpha_i$, $\beta$, $\beta_i$ are (optional) temporal and spatial annotations, the $C_j$'s are constraints and the $B_i$'s are atomic formulae. Constraints $C_j$ cannot be annotated. A *STACLP program* is a finite set of STACLP clauses.

## 3 Semantics of STACLP

In the definition of the semantics, without loss of generality, we assume all atoms to be annotated with `th`, `in`, `thr` or `inr` labels. In fact, `at` $t$ and `atp` $(x, y)$ annotations can be replaced with `th` $[t, t]$ and `thr` $[(x, x), (y, y)]$ respectively by exploiting the (`at th`) and (`atp thr`) axioms. Moreover, each atom in the object level program which is not two-annotated, i.e., which is labelled by at most one kind of annotation, is intended to be true throughout the whole lacking dimension(s). For instance an atom $A\,\texttt{thr}\,R$ is transformed into the two-annotated atom $A\,\texttt{thr}\,R\,\texttt{th}\,[0, \infty]$. Constraints remain unchanged.

The meta-interpreter for STACLP is defined by the following clauses:

$$demo(empty). \tag{1}$$

$$demo((B_1, B_2)) \leftarrow demo(B_1), demo(B_2) \tag{2}$$

$$demo(A\ \alpha\beta) \leftarrow \alpha \sqsubseteq \delta, \beta \sqsubseteq \gamma, clause(A\ \delta\gamma, B), demo(B) \tag{3}$$

$$\begin{aligned} demo(A\ \alpha'\beta') \leftarrow\ & \alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta, \alpha' \sqsubseteq \alpha, \beta' \sqsubseteq \beta, \\ & clause(A\ \alpha_1\beta_1, B), demo(B), demo(A\ \alpha_2\beta_2) \end{aligned} \tag{4}$$

$$demo(C) \leftarrow constraint(C), C \tag{5}$$

A clause $A\,\alpha\beta \leftarrow B$ of a STACLP program is represented at the meta-level by

$$clause(A\,\alpha\beta, B) \leftarrow valid(\alpha), valid(\beta) \tag{6}$$

where *valid* is a predicate that checks whether the interval or the region in the annotation is not empty.

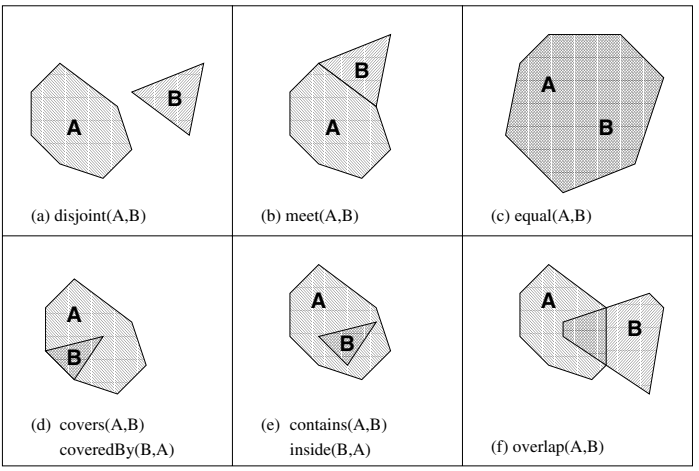



**Fig. 1.** Topological relations between two 2-dimensional objects

The first two clauses are the ordinary ones to solve the empty goal and a conjunction of goals. The resolution rule (clause (3)) implements both the Modus Ponens rule and the rule ($\sqsubseteq$). It states that given a clause $A\,\delta\gamma \leftarrow B$ whose body $B$ is solvable, we can derive the atom $A$ annotated with any annotation $\alpha\beta$ such that $\alpha \sqsubseteq \delta$ and $\beta \sqsubseteq \gamma$. Such constraints are processed by the constraint solver using the constraint theory for temporal and spatial annotations shown in § 2.2. Clause (4) implements the rule ($\sqcup$) (combined with Modus Ponens and rule ($\sqsubseteq$)). It states that if we can find a clause $A\,\alpha_1\beta_1 \leftarrow B$ such that the body $B$ is solvable, and if the atom $A$ can be proved with annotation $\alpha_2\beta_2$, then we can derive the atom $A$ labelled with any annotation less or equal than the least upper bound of $\alpha_1\beta_1$ and $\alpha_2\beta_2$. The constraint $\alpha_1\beta_1 \sqcup \alpha_2\beta_2 = \alpha\beta$ is solved by means of the axioms defining the least upper bound introduced in § 2.4. Clause (5) manages constraints by passing them directly to the constraint solver.

## 4 Qualitative Relations

Qualitative Spatial Reasoning has been recognized as a major point in the future developments of GIS [9]. In this section we prove that it is possible to represent topological and direction relations between spatial regions in STACLP. The idea is to express such relations by using simple inequalities between coordinate points. Given two rectangles their relationship can be found in constant time.

### 4.1 Topological Relations

The work by Egenhofer [7, 8] is the spatial counterpart of Allen's work on time intervals. He focuses on the class of *topological relationships* between spatial objects. A topological relation is a property invariant under homeomorphisms,

for instance it is preserved if the objects are translated or scaled or rotated. We restrict our attention to a space with only two dimensions and we present the 9-intersection model. Such a model is based on the intersection of the interior ($A^\circ$,$B^\circ$), the complement ($A^-$,$B^-$) and the boundary ($\delta A$,$\delta B$) of two 2-dimensional connected objects $A$ and $B$. Therefore a relation between $A$ and $B$ is represented by $R(A, B)$ as follows:

$$R(A,B) = \begin{pmatrix} \delta A \cap \delta B & \delta A \cap B^\circ & \delta A \cap B^- \\ A^\circ \cap \delta B & A^\circ \cap B^\circ & A^\circ \cap B^- \\ A^- \cap \delta B & A^- \cap B^\circ & A^- \cap B^- \end{pmatrix}$$

Each of these intersections can be empty or not. Of the $2^9$ possible different topological relations, only eight of these relations can be realized between two 2-dimensional objects and they are illustrated in Fig. 1.

### 4.2 Handling of Topological Relations in STACLP

Following Egenhofer's approach we define a predicate *topoRel* that determines the topological relation between two spatial rectangles by using the intersections among their interiors, exteriors and boundaries. For a rectangle $R$ we denote by $\circ R$ its interior, by $\delta R$ its boundary, and by $-R$ its exterior.

The predicate *topoRel* is defined by reflecting the definition of Egenhofer relation by using two predicates *intersect* and *no_intersect* denoting respectively whether the arguments of the predicates intersect or do not intersect.

$$topoRel(R_1, R_2, Topo_Relation) \leftarrow condition\ set$$

For instance, starting from the matrix $R(R_1, R_2)$ mentioned above, the *disjoint* topological relation is immediately encoded as:

$$\begin{aligned} topoRel(R_1, R_2, disjoint) \leftarrow\ & no_intersect(\delta R_1, \delta R_2),\ no_intersect(\delta R_1, \circ R_2), \\ & intersect(\delta R_1, -R_2),\ no_intersect(\circ R_1, \delta R_2), \\ & no_intersect(\circ R_1, \circ R_2),\ intersect(\circ R_1, -R_2), \\ & intersect(-R_1, \delta R_2),\ intersect(-R_1, \circ R_2), \\ & intersect(-R_1, -R_2) \end{aligned}$$

The remaining topological relations can be defined in a completely analogous way. Note that in this specific case, some of the predicates in the body are redundant and the clause can be simplified as $topoRel(R_1, R_2, disjoint) \leftarrow no_intersect(\delta R_1, \delta R_2),\ intersect(\delta R_1, -R_2)$.

The definitions of *intersect* and *no_intersect* express respectively when the interior, exterior or boundary of a region intersect or do not intersect the interior, exterior or boundary of another region. Such relations are given as constraints on the vertices of the rectangles. For example

$$\begin{aligned} intersect(\circ[(X_1, X_2), (Y_1, Y_2)], \circ[(X_3, X_4), (Y_3, Y_4)]) \leftarrow\ & X_1 < X_4, X_3 < X_2, \\ & Y_1 < Y_4, Y_3 < Y_2 \end{aligned}$$

$$intersect(\delta[(X_1, X_2), (Y_1, Y_2)], -[(X_3, X_4), (Y_3, Y_4)]) \leftarrow\ X_1 < X_3$$

$intersect(\delta[(X_1, X_2), (Y_1, Y_2)], -[(X_3, X_4), (Y_3, Y_4)]) \leftarrow \; X_4 < X_2$
$intersect(\delta[(X_1, X_2), (Y_1, Y_2)], -[(X_3, X_4), (Y_3, Y_4)]) \leftarrow \; Y_1 < Y_3$
$intersect(\delta[(X_1, X_2), (Y_1, Y_2)], -[(X_3, X_4), (Y_3, Y_4)]) \leftarrow \; Y_4 < Y_2$

*Example 1.* We want to select houses inside a park. Suppose that the park and houses are represented by the predicate *location* modelling their position.

$location(park)\ \texttt{thr}\ [(2,5),(20,50)].$ $\quad location(house1)\ \texttt{thr}\ [(2,2),(4,4)].$
$location(house2)\ \texttt{thr}\ [(4,7),(8,9)].$ $\quad location(house3)\ \texttt{thr}\ [(10,48),(12,50)].$

In order to ask the system which houses are inside the park we define the rule

$topoRelReg(Id1, Id2, Toporel) \leftarrow$
$\quad location(Id1)\ \texttt{thr}\ [(X_1, X_2), (Y_1, Y_2)], location(Id2)\ \texttt{thr}\ [(X_3, X_4), (Y_3, Y_4)],$
$\quad topoRel([(X_1, X_2), (Y_1, Y_2)], [(X_3, X_4), (Y_3, Y_4)], Toporel)$

This clause allows us to capture the relation between two objects knowing their identifiers. To get an answer to our request we ask $topoRelReg(park, X, inside)$, obtaining the answer $X = house2$ and $X = house3$.

### 4.3 Direction Relations

Direction relations deal with the orientation of spatial entities in space. There are no standard and universally recognized definitions of what a given direction relation is. For example, most people would agree that Germany is north of Italy, but what about France? Part of the country is north of Italy, part is west. We focus here on the model proposed by [16] where the direction relations are projection-based as we can see from Figure 2.

As we have done for the topological relation, we represent directions by means of a predicate, called *directionRel* that takes as arguments two rectangles and a direction among the eight described by the model.

$directionRel([(X_1, X_2), (Y_1, Y_2)], [(X_3, X_4), (Y_3, Y_4)], Direction_Rel) \leftarrow \ldots$

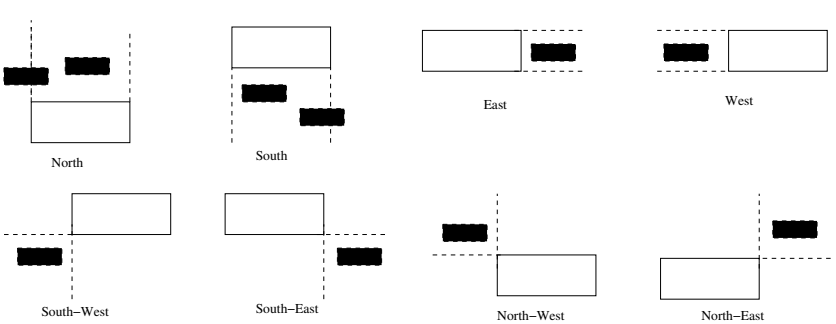



**Fig. 2.** Directions

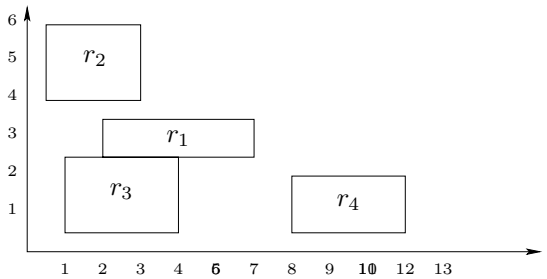



**Fig. 3.** Properties in the cooperative

As an example, consider the *south-west (sw)* direction. The rule below defines the constraints on the coordinates of the two rectangles that satisfy the *sw* relation (the second rectangle is in *sw* direction with respect to the first one).

$$directionRel([(X_1, X_2), (Y_1, Y_2)], [(X_3, X_4), (Y_3, Y_4)], sw) \leftarrow X_4 \leq X_1, Y_4 \leq Y_1$$

Again we use a more general rule that defines the direction relations among region identifiers instead of rectangles coordinates:

$$\begin{aligned} &directionRelReg(Id1, Id2, DirRel) \leftarrow \\ &\quad location(Id1)\ \texttt{thr}\ [(X_1, X_2), (Y_1, Y_2)], location(Id2)\ \texttt{thr}\ [(X_3, X_4), (Y_3, Y_4)], \\ &\quad directionRel([(X_1, X_2), (Y_1, Y_2)], [(X_3, X_4), (Y_3, Y_4)], DirRel) \end{aligned}$$

For instance, consider the query *Find all the towns "north of" Rome*. This query can be formulated as: $directionRelReg(IdTown, idRome, north)$, where *idRome* is the identifier of the town of Rome.

## 5 Example

Let us now present an application example that shows how the language can express qualitative spatial reasoning. We will use the topological and direction relations introduced in the previous section.

The context is an agricultural cooperative where the partners manage pieces of land owned by the cooperative itself. Each property is represented by a spatial object. Among other spatial analysis queries that can be modelled with our formalism, we focus here only on the qualitative aspects.

Assume that the properties of the cooperative are located as in Fig. 3. The query *Which is the binary topological relationship between* $r_1$ *and* $r_4$*?* is expressed by $topoRelReg(r_1, r_4, X)$. The answer is $X = disjoint$.

A query involving direction relations could be *Which is the direction relation between* $r_2$ *and* $r_4$*?* and it can be formulated as $directionRelReg(r_2, r_4, X)$, giving as answer $X = south - east$.

Suppose that a partner of the cooperative decides to buy another piece of land in order to enlarge his property. To do this, he has to check whether the owners of border properties are willing to sell their own land. Assume that the

partner owns property $r_3$. Now, the neighboring properties can be found by the query: $topoRelReg(r_3, X, meet)$.

In this context, our framework can be used also to identify the areas which can be devoted to the cultivation of specific vegetables. For example, suppose that, according to expert knowledge, a good growing area for corn depends on the kind of ground, the climate. Moreover corn should be preferably cultivated near potatoes because their sowing and gathering can be done at the same time.

$$\begin{aligned} cultivation_area(corn)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I} \leftarrow\ & favourable_area(corn)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I}, \\ & favourable_area(potatoes)\ \texttt{thr}\ \mathrm{S}\ \texttt{th}\ \mathrm{I}, \\ & topoRel(S, R, meet) \end{aligned}$$

The following rule expresses a different cultivation technique that could be performed by the cooperative: corn must be cultivated north of potatoes.

$$\begin{aligned} cultivation_area(corn)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I} \leftarrow\ & favourable_area(corn)\texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I}, \\ & favourable_area(potatoes)\texttt{thr}\ \mathrm{S}\ \texttt{th}\ \mathrm{I}, \\ & directionRel(S, R, north) \end{aligned}$$

The $favourable_area$ predicate is defined by a number of expert criteria such as the climate, the composition of the ground and the use of fertilizers.

$$\begin{aligned} favourable_area(X)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I} \leftarrow\ & favourable_climate(X)\texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I}, \\ & suitable_ground(X)\texttt{thr}\ \mathrm{R}\ \texttt{th}\ \mathrm{I} \end{aligned}$$

For instance, a favourable area for corn should have a temperature around 10 degree during March and April and should be fertilized in February. Then the area could be devoted to corn from March (sowing) to September (gathering). This is expressed by the rule:

$$\begin{aligned} favourable_area(corn)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ [\mathrm{mar,sep}] \leftarrow\ & temp(X)\ \texttt{thr}\ \mathrm{R}\ \texttt{th}\ [\mathrm{mar,apr}], \\ & X > 9,\ X < 14, \\ & fertilized\ \texttt{thr}\ \mathrm{R}\ \texttt{in}\ [\mathrm{feb,feb}] \end{aligned}$$

Despite its simplicity this example shows how this approach is particularly suited to represent in a declarative, intuitive and compact way, forms of qualitative spatial reasoning.

## 6 Conclusions

We presented a spatio-temporal language STACLP which can support both quantitative and qualitative spatio-temporal reasoning. In particular we focused our attention on *spatial qualitative reasoning*, showing how topological and direction relations between rectangular regions can be expressed in our formalism.

Our current activity consists of extending such an approach to generic regions. It is worth noticing that in our setting, a non rectangular region can be represented (possibly in an approximated way) as a union of rectangles. A region

*idreg*, divided into $n$ rectangles $\{[(x_1^i, x_2^i), (y_1^i, y_2^i)] \mid i = 1, \ldots, n\}$, is modelled by a collection of unit clauses as follows:

$$region(idreg)\ \texttt{thr}\ [(x_1^1, x_2^1), (y_1^1, y_2^1)]. \quad \ldots \quad region(idreg)\ \texttt{thr}\ [(x_1^n, x_2^n), (y_1^n, y_2^n)].$$

Unfortunately, it is not possible to determine the topological relation between two generic regions checking the relations between their rectangular components [14]. Following the line suggested by Papadias and Theodoridis [16] the treatment of generic regions can be coped with a two steps process. In the first step spatial objects are approximated with Minimum Bounding Rectangles (MBR) and the topological relations between such MBRs are determined in order to rapidly eliminate objects that surely do not satisfy the query. After this filtering step, by using computational geometry techniques, each candidate is examined in order to detect and eliminate false hits.

## Acknowledgments

We thank Enrico Orsini who collaborated at a preliminary stage of this work, Thom Frühwirth for his comments, the anonymous referees for their useful suggestions and Paolo Baldan for his careful reading. This work has been partially supported by European Project IST-1999-14189 - Rev!gis.

## References

[1] COSIT - Conference On Spatial Information Theory. Volume 2205 of *Lecture Notes in Computer Science*. Springer, 2001. 78

[2] D. Aquilino, P. Asirelli, A. Formuso, C. Renso, and F. Turini. Using MedLan to Integrate Geographical Data. *Journal of Logic Programming*, 43(1):3–14, 2000. 78

[3] B. L. Clarke. A calculus of individuals based on 'connection'. *Notre Dame Journal of Formal Logic*, 22(3):204–218, 1981. 79

[4] E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships for End-User Interaction. In *Advances in Spatial Databases*, volume 692 of *LNCS*, pages 277–295, 1993. 79

[5] A. G. Cohn and S. M. Hazarika. Qualitative spatial representation and reasoning: an overview. *Fundamenta Informaticae*, 45:1–29, 2001. 78

[6] OpenGIS Consortium. OpenGIS Simple Features Specification For OLE/COM, 1999. http://www.opengis.org/techno/specs/99-050.pdf. 79

[7] M. J. Egenhofer. Reasoning about binary topological relations. In *Advances in Spatial Databases*, volume 525 of *LNCS*, pages 143–160. Springer, 1991. 78, 79, 84

[8] M. J. Egenhofer. User interfaces. In *Cognitive Aspects of Human-Computer Interaction for Geographical Information Systems*, pages 1–8. Kluwer Academic, 1995. 78, 84

[9] M. J. Egenhofer and D. Mark. Naive geography. In *COSIT 95*, volume 988 of *LNCS*, pages 1–15. Springer, 1995. 78, 84

[10] A. Frank. Qualitative spatial reasoning: Cardinal directions as an example. *International Journal of Geographical Information Science*, 10(3):269–290, 1996. 79

[11] T. Frühwirth. Temporal Annotated Constraint Logic Programming. *Journal of Symbolic Computation*, 22:555–583, 1996. 79, 82

[12] R. K. Goyal. Similarity assessment for cardinal directions between extended spatial objects. Technical report, The University of Maine, 2000. PhD Thesis. 79

[13] R. K. Goyal and M. J. Egenhofer. Cardinal directions between extended spatial objects. *IEEE Transactions on Knowledge and Data Engineering*. In press. 79

[14] E. Orsini. Ragionamento Spazio-Temporale basato su logica, vincoli e annotazioni. Master's thesis, Dipartimento di Informatica, Università di Pisa, 2001. 89

[15] D. Papadias, T. Sellis, Y. Theodoridis, and M. J. Egenhofer. Topological relations in the world of minimum bounding rectangles: a study with R-trees. In *ACM SIGMOD Int. Conf. on Management of Data*, pages 92–103, 1995. 79

[16] D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997. 78, 79, 86, 89

[17] D. Peuquet and C.-X. Zhan. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, 20(1):65–74, 1987. 79

[18] A. Raffaetà and T. Frühwirth. Spatio-temporal annotated constraint logic programming. In *PADL2001*, volume 1990 of *LNCS*, pages 259–273. Springer, 2001. 78, 79, 82

[19] D. A. Randell, Z. Cui, and A. G. Cohn. A spatial logic based on region and connection. In *Principles of Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, 1992. 78

[20] K. Zimmermann and C. Freksa. Qualitative Spatial Reasoning Using Orientation, Distance, and Path Knowledge. *Applied Intelligence*, 6:46–58, 1996. 79

# On Managing Temporal Information for Handling Durative Actions in LPG

Alfonso Gerevini, Alessandro Saetti, and Ivan Serina

Dipartimento di Elettronica per l'Automazione
Università degli Studi di Brescia
via Branze 38, 25123 Brescia, Italy
{gerevini,saetti,serina}@ing.unibs.it

**Abstract.** LPG is a recent planner based on local search and planning graphs which supports durative actions specified by the new standard language PDDL2.1. The planner was awarded at the 3rd Planning Competition (Toulouse, 2002) for its very good performance. This paper focuses on how LPG represents and manages temporal information to handle durative actions during the construction of a plan. In particular, we introduce a plan representation called *Temporal Durative Action Graph* (TDA-graph) which distinguishes different types of constraints for ordering the actions and allows to generate plans with a high degree of parallelism. An experimental analysis shows that the techniques here presented are effective, and that in temporal domains our planner outperforms all other fully-automated planners that took part in the contest.

## 1 Introduction

Local search is emerging as a powerful method for domain-independent planning (e.g., [4, 8]). A first version of LPG, presented in [4, 6], uses local search in the space of *action graphs* (A-graphs), particular subgraphs of the planning graph representation [1]. This version handled only STRIPS domains possibly extended with simple costs associated with the actions. In this paper we present some extensions to handle planning domains specified in the recent PDDL2.1 language supporting "durative actions" [2] . In particular, we introduce a plan representation called *temporal durative action graph* (TDA-graph) that is currently used by LPG to handle durative actions and the temporal information associated with them. Essentially, durative actions relax the (strong) STRIPS assumption that actions are instantaneous. This allows to specify different types of preconditions, depending on when they are required to be true (either at the beginning of the action, at its end, or over all its duration). Similarly for the effects, which can become true either at the beginning or at the end of a durative action.

Like traditional planning graphs, TDA-graphs are directed acyclic "levelled" graphs with two kinds of nodes (action nodes and fact nodes). Moreover, in TDA-graphs action nodes are marked with temporal values estimating the earliest time when the corresponding actions terminate. Similarly, fact nodes are marked

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 91–104, 2003.
© Springer-Verlag Berlin Heidelberg 2003

with temporal values estimating the earliest time when the corresponding facts becomes true. Finally, a set of ordering constraints is maintained during search to handle mutually exclusive actions (called *mutex* actions), and to take account of the "causal" relations in the current plan.

LPG is an incremental planner based on stochastic local search that can compute a succession of plans, each of which improves the quality of the previous ones according to a plan metric specified in the planning domain. In the 3rd International Planning Competition (IPC) [3] our planner showed excellent performance on a large set of problems, in terms of both speed to compute the first solution and quality of the best solution that can be computed by the incremental process. At the time of the competition these extensions had been only partially integrated into LPG. In particular, all mutex actions are handled by imposing an ordering constraint between the *end* of an action and the *start* of the other (i.e., no overlapping between mutex actions is permitted). The new version can handle different type of mutex actions (depending on the type of precondition/effect that generates the interference between the actions making them mutually exclusive). Different types interferences are handled by imposing different types of ordering constraints, which can permit action overlapping, and hence can lead to plans of better quality (i.e., of shorter makespan).

In [6, 5] we present a collection of search techniques used by LPG. In this paper we concentrate on the how temporal information are represented and managed for handling durative actions. Section 2 presents the TDA-graph plan representation; section 3 describes the different types of ordering constraints used by LPG, how they are stated by the planner, and how they are used to compute the temporal values associated with the actions of the plan; section 4 gives the results of an experimental analysis with some domains used in the 3rd IPC; finally, section 5 gives our conclusions.

## 2 Plan Representation

In our approach, plans are represented through *action graphs* [4, 6], i.e. particular subgraphs of the planning graph representation [1]. In the following we will assume that the reader is familiar with the planning graph representation and with the related terminology. Given a planning graph $\mathcal{G}$ for a planning problem, it's possible to assume that the goal nodes of $\mathcal{G}$ in the last level represent the preconditions of a special action $a_{end}$, which is the last action in any valid plan, while the fact nodes of the first level represent the effects of a special action $a_{start}$, which is the first action in any valid plan.

An *action graph* (A-graph) for $\mathcal{G}$ is a subgraph $\mathcal{A}$ of $\mathcal{G}$ such that, if $a$ is an action node of $\mathcal{G}$ in $\mathcal{A}$, then also the fact nodes of $\mathcal{G}$ corresponding to the preconditions and positive effects of $a$ are in $\mathcal{A}$, together with the edges connecting them to $a$. An action graph can contain some *inconsistencies*, i.e., an action with precondition nodes that are not *supported*, or a pair of action nodes involved in

```
(:durative-action calibrate
 :parameters (?s - satellite ?i - instrument ?d - direction)
 :duration (= ?duration 5)
 :condition (and (over all (on_board ?i ?s)) (over all (calibration_target ?i ?d))
      (at start (pointing ?s ?d)) (over all (power_on ?i)) (at end (power_on ?i)))
 :effect (at end (calibrated ?i)))

(:durative-action turn_to
 :parameters (?s - satellite ?d_new - direction ?d_prev - direction)
 :duration (= ?duration 5)
 :condition (and (at start (pointing ?s ?d_prev)) (over all (not (= ?d_new ?d_prev))))
 :effect (and (at end (pointing ?s ?d_new)) (at start (not (pointing ?s ?d_prev)))))
```

**Fig. 1.** Two durative operators in the temporal variants of the "Satellite" domain [3]

a *mutex relation*.[1] In general, a goal $g$ or a precondition node $q$ at a level $i$ is supported in an action graph $\mathcal{A}$ of $\mathcal{G}$ if either (i) in $\mathcal{A}$ there is an action node (or a no-op node) at level $i-1$ representing an action with (positive) effect $q$, or (ii) $i = 1$ (i.e., $q$ is a proposition of the initial state). An action graph without inconsistencies represents a valid plan and is called *solution graph*. A *solution graph* for $\mathcal{G}$ is an action graph $\mathcal{A}_s$ of $\mathcal{G}$ where all precondition nodes of its action nodes are supported, and there are no mutex relations between its action nodes.

The version of LPG that took part in the 3rd IPC uses a particular subset of the action graphs, called *linear action graphs with propagation* [7]. A *linear action graph with propagation (LA-graph)* of $\mathcal{G}$ is an A-graph of $\mathcal{G}$ in which each action level contains at most one action node and any number of no-op nodes, and such that if $a$ is an action node of $\mathcal{A}$ at level $l$, then, for any positive effect $e$ of $a$ and any level $l' > l$ of $\mathcal{A}$, the no-op node of $e$ at level $l'$ is in $\mathcal{A}$, unless there is another action node at a level $l''$ ($l < l'' \leq l'$) which is mutex with the no-op node.[2] In a linear action graph with propagation, the unsupported preconditions are the only type of inconsistencies that the search process needs to handle explicitly. Since in the rest of the paper we consider only linear action graphs with propagation, we will abbreviate their name simply to linear action graphs (leaving implicit that they include the no-op propagation).

The current version of LPG can handle levels 2 and 3 of PDDL2.1. Level 2 introduces numerical quantities and level 3 a new model of actions, called *durative actions*, that supports stronger parallelism among the actions in the plan obtained by distinguishing different ways in which they can overlap [2]. This paper focuses on the representation of durative actions. We indicate with $Cond_S(a)$, $Cond_O(a)$ and $Cond_E(a)$ the `at start`, `over all` and `at end` conditions, respectively, of an action $a$; with $Eff_S(a)$ and $Eff_E(a)$ the `at start` and `at end` effects, respectively, of $a$; with $Add_S(a)/Del_S(a)$ the `at start` ad-

[1] LPG considers only pairs of actions *globally mutex* [7], i.e. that hold at every level of $\mathcal{G}$.

[2] As noted in [5], having only one action in each level of a LA-graph does not prevent the generation of parallel (partially ordered) plans.

ditive/delete effects of $a$; with $Add_E(a)$/ $Del_E(a)$ the `at end` additive/delete effects of $a$, and with $dur(a)$ the duration of $a$. Figure 1 shows two operators of the temporal variants of the "Satellite" domain used in the 3rd IPC. For instance the facts `(pointing satellite0 station2)` and `(not (= phen4 station2))` belong to the $Cond_S$ and $Cond_O$ sets of the action `(turn_to satellite0 phen4 station2)`, respectively. The facts `(pointing satellite0 phen4)` and `(not (pointing satellite0 station2))` are achieved at the end and at the beginning of the action, respectively, so they belong to the $Add_E$ and $Del_S$ sets of the previous action `turn_to`, respectively.

In simple STRIPS domains, the additive effects of an action $a$ at a level $l$ are represented by fact nodes at the level $l+1$, and its preconditions by fact nodes at the level $l$. For PDDL2.1 domains involving durative actions, in order to represent the facts that become true after the beginning of the action $a$, we could introduce a third level between $l$ and $l+1$. Instead, LPG uses no-op nodes to represent the state of the world during the execution of an action. The `at start` additive/delete effects of $a$ at level $l$, $Add_S(a)/Del_S(a)$, are achieved after the beginning of $a$, and so LPG introduces/removes the corresponding no-op nodes at level $l$ of $\mathcal{A}$. The `at end` additive/delete effects of $a$, $Add_E(a)/Del_E(a)$, are achieved at the end of $a$, and so they do not affect any no-op node at level $l$; LPG introduces/removes the fact nodes of $Add_E(a)/Del_E(a)$ at level $l+1$ of $\mathcal{A}$. The `at start` conditions of $a$, $Cond_S(a)$, must be achieved at the beginning of $a$, and so LPG verifies that the corresponding fact nodes at level $l$ are supported in $\mathcal{A}$. The `over all` conditions of $a$, $Cond_O(a)$, must be achieved during the full duration of the execution of $a$, and so LPG verifies that the corresponding no-op nodes are supported at level $l$. The `at end` conditions of $a$, $Cond_E(a)$, must be true at the end of $a$; more precisely, they must be achieved after the `at start` effects become true and before the `at end` effects become true. Therefore, as for the `over all` conditions, LPG checks that the no-op nodes corresponding to the `at end` conditions are supported at level $l$. The difference between the `over all` and `at end` conditions consists in a different temporal management. This additional way of using the no-op nodes in action graphs leads to the definition of a new class of action graphs called *durative action graphs*.

**Definition 1.** *A* **Durative Action Graph** *(DA-graph) for $\mathcal{G}$ is a linear subgraph $\mathcal{A}$ of $\mathcal{G}$ such that, if $a$ is an action node of $\mathcal{G}$ in $\mathcal{A}$ at level $l$, then*

- *the fact nodes corresponding to the* `at start` *conditions of $a$ are in $\mathcal{A}$ at level $l$,*
- *the fact nodes corresponding to the* `at end` *effects of $a$ are in $\mathcal{A}$ at level $l+1$,*
- *the no-op nodes corresponding to the* `over all` *conditions of $a$, the* `at end` *conditions of $a$, and the* `at start` *effects of $a$ are in $\mathcal{A}$ at level $l$,*
- *all edges connecting the nodes of the previous items to $a$ are in $\mathcal{A}$.*

LPG represents durative actions by modifying the original structure of the planning graph $\mathcal{G}$, i.e., by introducing edges from action nodes to no-op nodes at the same level of the graph to represent `at start` effects, and edges from no-op nodes to action nodes to represent `at end` and `over all` conditions.

In order to represent the temporal information associated with the end points of an action, our planner (i) assigns real values to action, fact and no-op nodes of the DA-graph, and (ii) uses a set $\Omega$ of ordering constraints between action nodes. The value associated with a fact or no-op node $f$ represents the (estimated) earliest time at which $f$ becomes true, while the value associated with an action node $a$ represents the (estimated) earliest time when the execution of $a$ can terminate. These estimates are derived from the duration of the actions in the DA-graph and the ordering constraints between them that are stated in $\Omega$. Obviously, the value associated with $a_{start}$ is zero, while the value associated with $a_{end}$ represents the makespan of the current plan. This assignment of real values to the durative action graph nodes leads to the representation used by LPG to handle durative actions called *temporal durative action graph* (an example of those graphs will be given in section 3.2).

**Definition 2.** *A* **Temporal Durative Action Graph** *(TDA-graph) of $\mathcal{G}$ is a triple $\langle \mathcal{A}, \mathcal{T}, \Omega \rangle$ where $\mathcal{A}$ is a durative action graph; $\mathcal{T}$ is an assignment of real values to the fact, no-op and action nodes of $\mathcal{A}$; $\Omega$ is a set of ordering constraints between action nodes of $\mathcal{A}$.*

## 3 Temporal Management for Durative Actions

In this session we discuss different types of ordering constraints in the set $\Omega$ of TDA-graphs. Moreover, we describe how and when the temporal values of $\mathcal{T}$ are computed.

### 3.1 Ordering Constraints

The original planning graph representation [1] imposes the global constraint that, for any action $a$ at any level $l$ of the graph, every action at the following level starts after the end of $a$. However, this can impose some unnecessary ordering constraint between actions of different levels which could limit action parallelism in the plan (and hence its quality in terms of makespan). TDA-graphs support a more flexible representation by handling ordering constraints explicitly: actions are ordered only through the ordering constraints in $\Omega$ that are stated by the planner during search. At each step of the search LPG adds/removes an action $a$ to/from the current TDA-graph. When $a$ is added, LPG generates (i) appropriate "causal links" between $a$ and other actions with preconditions achieved by $a$, (ii) ordering constraints between $a$ and every action in the TDA-graph that is mutex with $a$ [7].

For simple traditional domains where every condition is of type `over all` and any effect is of type `at end`, the ordering constraints in $\Omega$ are of two types: constraints between actions that are implicitly ordered by the causal structure of the plan ($\prec_C$*-constraints*), and constraints imposed by the planner to deal with mutually exclusive actions ($\prec_E$*-constraints*). $a \prec_C b$ belongs to $\Omega$ if and only if $a$ is used to achieve a condition node of $b$ in $\mathcal{A}$, while $a \prec_E b$ (or $b \prec_E a$)

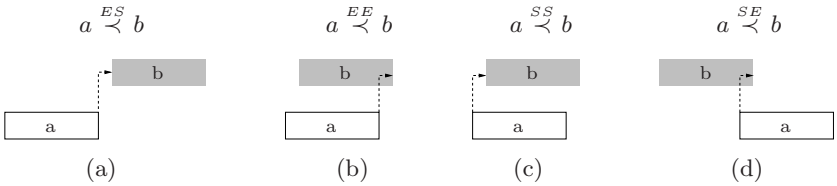



**Fig. 2.** Types of ordering constraints between durative actions

belongs to $\Omega$ only if $a$ and $b$ are mutually exclusive in $\mathcal{A}$. If $a$ and $b$ are mutex actions, the planner appropriately imposes either $a \prec_E b$ or $b \prec_E a$. LPG chooses $a \prec_E b$ if the level of $a$ precedes the level of $b$, $b \prec_E a$ otherwise. Under this assumption on the "direction" in which $\prec_E$*-constraints* are imposed, it is easy to see that the levels of the graph correspond to a topological order of the actions in the represented plan satisfying every ordering constraint in $\Omega$. An ordering constraint $a \prec b$ in $\Omega$ (where "$\prec$" stands for $\prec_C$ or $\prec_E$) states that the beginning of $b$ comes after the end of $a$. Our planner schedules actions in a way that the execution of an action is anticipated as soon as possible; and so $a \prec b$, means that $b$ starts immediately after the end of $a$.

**Table 1.** Ordering constraints between the durative actions $a$ and $b$, according to the possible casual relations ($\prec_C$) and to the *mutex* between conditions and effects of $a$ and $b$ ($\prec_E$). The label $\prec_E$ marking an entry of the table indicates that at least a proposition of the set associated with the raw of the entry is *mutex* with at least a proposition of the set associated with the column of the entry. The label $\prec_C$ marking an entry of the table indicates that at least one proposition of the set associated with the raw of the entry supports a proposition of the set associated with the column of the entry

| $\overset{a}{\prec_d}$ | $Cond_S(b)$ | $Cond_O(b)$ | $Cond_E(b)$ | $Eff_S(b)$ | $Eff_E(b)$ |
|---|---|---|---|---|---|
| $Cond_S(a)$ | 1 $\prec_E$ | 2 $\prec_E$ | 3 $\prec_E$ | 4 $\prec_E$ | 5 $\prec_E$ |
| $Cond_O(a)$ | 6 $\prec_E$ | 7 $\prec_E$ | 8 $\prec_E$ | 9 $\prec_E$ | 10 $\prec_E$ |
| $Cond_E(a)$ | 11 $\prec_E$ | 12 $\prec_E$ | 13 $\prec_E$ | 14 $\prec_E$ | 15 $\prec_E$ |
| $Eff_S(a)$ | **16** $\prec_C, \prec_E$ | 17 $\prec_C, \prec_E$ | 18 $\prec_C, \prec_E$ | 19 $\prec_E$ | 20 $\prec_E$ |
| $Eff_E(a)$ | 21 $\prec_C, \prec_E$ | 22 $\prec_C, \prec_E$ | 23 $\prec_C, \prec_E$ | 24 $\prec_E$ | 25 $\prec_E$ |

For more complex domains, where operators can contain all types of conditions and effects for a durative action, LPG distinguishes some additional types of ordering constraints in $\Omega$. In fact, two ordered actions $a$, $b$ can overlap in four different ways, depending on the type of conditions and effects involved in the casual relation between $a$ and $b$, or in the mutex relation between them. The four orderings constraint different pairs of end points of the intervals associated at the duration of the actions. We indicate these constraints with $\overset{ES}{\prec}$, $\overset{EE}{\prec}$, $\overset{SS}{\prec}$ and $\overset{SE}{\prec}$.[3] $a \overset{ES}{\prec} b$ belongs to $\Omega$ if $a \prec_C b \in \Omega$ or $a \prec_E b \in \Omega$ and $b$ can not start before the end of $a$ (see case $a$ of figure 2). $a \overset{EE}{\prec} b$ belongs to $\Omega$ if $a \prec_C b \in \Omega$ or $a \prec_E b \in \Omega$ and $b$ can not end before the end of $a$ (see case $b$ of figure 2). $a \overset{SS}{\prec} b$ belongs to $\Omega$ if $a \prec_C b \in \Omega$ or $a \prec_E b \in \Omega$ and $b$ can not start before the beginning of $a$ (see case $c$ of figure 2). $a \overset{SE}{\prec} b$ belongs to $\Omega$ if $a \prec_C b \in \Omega$ or $a \prec_E b \in \Omega$ and $b$ can not end before the beginning of $a$ (see case $d$ of figure 2).

Table 1 shows all possible situations that generate in $\Omega$ one of the ordering constraints of figure 2. For example, the sixteenth entry of the table 1 shows that $a \overset{SS}{\prec} b \in \Omega$ if (i) at least an effect of type `at start` of $a$ supports a condition of type `at start` of $b$ ($\overset{SS}{\prec}_C$*-constraints*), or (ii) at least an effect of type `at start` of $a$ is mutex with a condition of type `at start` of $b$ ($\overset{SS}{\prec}_E$*-constraints*).

In general, if $a \prec b$, there is at least one ordering constraint in $\Omega$ between $a$ and $b$ of type $\overset{EE}{\prec}$, $\overset{ES}{\prec}$, $\overset{SE}{\prec}$, or $\overset{SS}{\prec}$. If there are more than one ordering constraints between $a$ and $b$, it is possible to simplify $\Omega$ by removing all them from $\Omega$, except the strongest one, i.e., the constraint for which the execution of $b$ is most delayed. The strongest constraint is $\overset{ES}{\prec}$, because $a \overset{ES}{\prec} b$ imposes that the execution of $a$ can not overlap the execution of $b$, i.e., $b$ can not start before the end of $a$. The weakest constraint is $\overset{SE}{\prec}$, because $a \overset{SE}{\prec} b$ imposes that $b$ ends after the beginning of $a$; so, $b$ can start before the beginning of $a$. Note that the strongest constraint between $\overset{SS}{\prec}$ and $\overset{EE}{\prec}$ depends on the durations of the actions involved. In particular, if the duration of $b$ is longer than the duration of $a$, then $\overset{SS}{\prec}$ is stronger than $\overset{EE}{\prec}$; if the duration of $b$ is shorter than the duration of $a$, then $\overset{EE}{\prec}$ is stronger than $\overset{SS}{\prec}$ (see figure 3). If $a$ and $b$ have the same duration, the constraint $\overset{SS}{\prec}$ is as strong as $\overset{EE}{\prec}$. Finally, when the duration of at least one of the actions involved depends on the particular state of the world $S$ in which the action is applied, if $a \overset{ES}{\prec} b \notin \Omega$ but $a \{\overset{SS}{\prec}, \overset{EE}{\prec}\} b \in \Omega$, then both $a \overset{SS}{\prec} b$ and $a \overset{EE}{\prec} b$ must be kept in $\Omega$ and evaluated at "runtime". This is because the state $S$ is only partially defined in the current partial plan, and it could change when the plan is modified. An example of a domain where action duration depends on the state in which the action is applied is the "Time" variant of "Rover" [3]. In this

[3] There is a fifth possible ordering constraint between two durative actions $a$ and $b$; e.g., if $dur(b) > dur(a)$, it is possible that $a$ supports an `at end` condition of $b$ and $b$ supports or deletes an `at start` condition of $a$; but currently LPG does not consider this case.

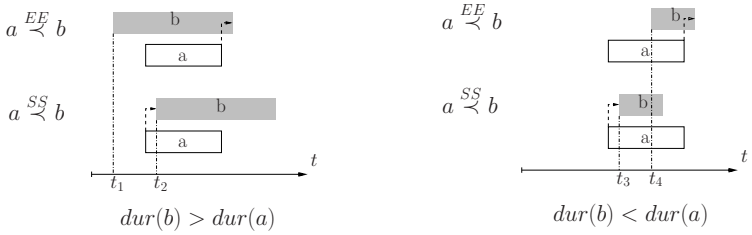



**Fig. 3.** The strongest ordering constraints between $\overset{SS}{\prec}$ and $\overset{EE}{\prec}$ depending on the durations ($dur$) of the actions $a$ and $b$

domain the duration of the operator "`recharge`" depends on the level of energy currently available for the rover.[4]

### 3.2 Temporal Values of TDA-graph Nodes

The constraints stored in $\Omega$ are useful to compute the temporal values of the nodes in a TDA-graph. These values are updated each time an action node is added or removed, which in the worst case requires linear time with respect to the number of nodes in the graph [7]. We denote with $Time(x)$ the temporal value assigned by $\mathcal{T}$ to a node $x$. In domains where all effects are of type `at end` and conditions of type `over all`, LPG computes the temporal value of an action $b$ by simply examining the maximal values over the temporal values of the actions $a$ in $\mathcal{A}$ that must precede $b$ according to $\Omega$:

$$Time(b) = \max_{a \prec b \in \Omega} \big\{Time(a), 0\big\} + dur(b) + \epsilon.^{5}$$

If there is no action node that must precede $a$ according to $\Omega$, then $b$ can not start before zero; so, $Time(b)$ is set to the duration of $b$.[6] For more general TDA-graphs supporting every type of conditions/effects described in section 3.1, the definition of $Time(b)$ is more complex because it takes account of all types of ordering constraints previously introduced:

$$Time(b) = \Bigg[\max\Bigg\{\max_{a \overset{SE}{\prec} b \in \Omega} \big\{Time(a) - dur(a)\big\} - dur(b), \max_{a \overset{ES}{\prec} b \in \Omega} \big\{Time(a)\big\},$$
$$\max_{a \overset{EE}{\prec} b \in \Omega} \big\{Time(a)\big\} - dur(b), \max_{a \overset{SS}{\prec} b \in \Omega} \big\{Time(a) - dur(a)\big\}, 0\Bigg\}\Bigg] + dur(b) + \epsilon.$$

[4] More precisely, the duration is defined by the following expression: `:duration (= ?duration (/ (- 80 (energy ?x)) (recharge-rate ?x)))`, where `?x` indicates the rover.

[5] LPG introduces a positive $\epsilon$ quantity to satisfy the ordering constraints. Without this term the beginning of $b$ would be equal to the end of an action $a$ that must precede $b$.

[6] In order to give a better estimate of the temporal value at which an action terminates, if a condition is not supported, instead of zero, LPG estimates the earliest temporal value at which the corresponding proposition becomes true, as described in [7].

The term into square brackets represents the earliest temporal value at which the execution of $b$ can start, in accordance with the ordering constraints of type $\overset{SE}{\prec}$, $\overset{ES}{\prec}$, $\overset{EE}{\prec}$ and $\overset{SS}{\prec}$ involving $b$ that are present in the current TDA-graph.

The temporal values of the action nodes are used to compute the temporal values of the fact and no-op nodes. If a fact $f$ is supported by more actions, LPG considers the temporal value of the action that supports $f$ earlier. If all conditions are of type `over all` and effects of type `at end`, LPG computes the temporal value of a fact node $f$ by simply examining the minimum values over the temporal values of the actions $a$ in $\mathcal{A}$ that support $f$:

$$Time(f) = \min_{a \in \Lambda(f)} \{Time(a)\},$$

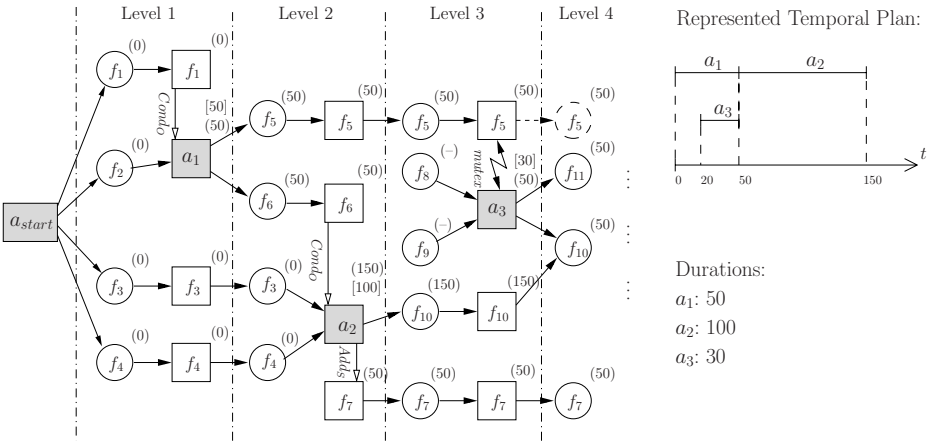
where $\Lambda(f)$ is the set of action nodes that support the fact node $f$. More in general, LPG distinguishes the cases in which $f$ is supported at the beginning or at the end of an action, and so the temporal value of a fact node $f$ is computed according to the following more complex definition of $Time(f)$:

$$Time(f) = \min \left\{ \min_{a \in \Lambda_E(f)} \{Time(a)\}, \min_{a \in \Lambda_S(f)} \{Time(a) - dur(a)\} \right\},$$

where $\Lambda_E(f)$ and $\Lambda_S(f)$ are the sets of action nodes that support $f$ at the end and at the beginning, respectively, of the corresponding actions.

In planning problems where it is important to minimize the makespan of the plan, LPG uses these temporal values to guide the search toward a direction that improves the quality of the plan under construction. In particular, they are used to estimate the temporal value at which a condition not supported could become supported [5].

Figure 4 gives an example of a portion of a TDA-graph containing four action nodes ($a_{1\ldots3,\,start}$) and several fact and no-op nodes representing eleven facts ($f_{1\ldots11}$). Since $a_1$ supports an `over all` condition of $a_2$, $a_1 \overset{ES}{\prec}_C a_2$ belongs to $\Omega$. $a_1 \overset{EE}{\prec}_E a_3$ belongs to $\Omega$ because an `at end` effect of $a_1$ is mutex with an `at end` effect of $a_3$. $a_{start} \overset{ES}{\prec}_C a_1 \in \Omega$ because $f_1$, that is an `over all` condition of $a_1$, and $f_2$, that is an `at start` condition of $a_1$, belong to the initial state. Similarly, $a_{start} \overset{ES}{\prec}_C a_2 \in \Omega$ because $f_3$ and $f_4$, that are `at start` conditions of $a_2$, belong to the initial state. The temporal value assigned to facts $f_{1\ldots4}$ at the first level is zero, because they belong to the initial state. Since $a_{start} \overset{ES}{\prec}_C a_1 \in \Omega$, $Time(a_1)$ is the sum of $Time(a_{start})$ and of the duration of $a_1$, i.e., $0 + 50$. $f_5$ and $f_6$ belong to $Add_E(a_1)$; so, the time assigned to the fact nodes $f_5$ and $f_6$ at level 1 is equal to $Time(a_1)$ (the end time of $a_1$). $Time(a_2)$ is given by the sum of the duration of $a_2$ and the maximum over $Time(a_{start})$ and $Time(a_1)$, because $\{a_1, a_{start}\} \overset{ES}{\prec}_C a_2 \in \Omega$, i.e., $50 + 100$. $f_{10}$ is an `at end` effect of $a_2$, and so the time assigned to $f_{10}$ at level 3 is equal to $Time(a_2)$. Since $f_7 \in Add_S(a_2)$, the time assigned to the no-op node $f_7$ at level 2 is equal to $Time(a_2) - dur(a_2)$ (the start time of $a_2$). Since $a_1 \overset{EE}{\prec}_E a_3 \in \Omega$, $Time(a_3)$ is given by the sum of the




$$\Omega = \{a_{start} \overset{ES}{\prec}_C a_1,\ a_1 \overset{ES}{\prec}_C a_2,\ a_{start} \overset{ES}{\prec}_C a_2,\ a_1 \overset{EE}{\prec}_E a_3\}$$

**Fig. 4.** A portion of a TDA-graph. Circle nodes are fact nodes; gray square nodes are action nodes, while the other square nodes are no-op nodes. The edges from no-op nodes to action nodes and from action nodes to no-op nodes represent the `over all` conditions and `at start` effects, respectively. Round brackets contain temporal values assigned by $\mathcal{T}$ to fact nodes and action nodes. The numbers in square brackets represent action durations. "(–)" indicates that the corresponding fact node is not supported

duration of $a_3$ and the maximum between zero (because conditions $f_8$ and $f_9$ are not supported) and $Time(a_1) - dur(a_3)$, i.e., $30 + 50 - 30$. $f_{11}$ at level 4 is supported only by $a_3$ at the end of it; therefore, the temporal value associated with $f_{11}$ is equal to $Time(a_3)$. $f_{10}$ at level 4 is supported at the end of $a_2$ and $a_3$. But since $Time(a_2) > Time(a_3)$, we have that $Time(f_{10})$ at level 4 is equal to $Time(a_3)$.

## 4 Experimental Results

In this section we present some experimental results illustrating the efficiency of LPG in the domains involving durative actions that were used in the 3rd IPC (i.e., "SimpleTime", "Time" and "Complex").[7] The results of LPG correspond to

[7] The system is available at `http://prometeo.ing.unibs.it/lpg`. For a description of these domains and of the relative variants the reader may see the official web site of the 3rd IPC (`www.dur.ac.uk/d.p.long/competition.html`). Detailed results for all the problem tested are available at the web site of LPG.

```
0.001:   (TURN_TO SATELLITE0 STATION2 PHEN6)[5.000]
0.002:   (SWITCH_ON INSTR0 SATELLITE0)[2.000]
5.003:   (CALIBRATE SATELLITE0 INSTR0 STATION2)[5.000]
5.004:   (TURN_TO SATELLITE0 PHEN4 STATION2)[5.000]
10.005:  (TAKE_IMAGE SATELLITE0 PHEN4 INSTR0 THERMOGRAPH0)[7.000]
17.006:  (TURN_TO SATELLITE0 PHEN6 PHEN4)[5.000]
22.007:  (TAKE_IMAGE SATELLITE0 PHEN6 INSTR0 THERMOGRAPH0)[7.000]
29.008:  (TURN_TO SATELLITE0 STAR5 PHEN6)[5.000]
34.009:  (TAKE_IMAGE SATELLITE0 STAR5 INSTR0 THERMOGRAPH0)[7.000]
```

**Fig. 5.** Example of a simple plan in the "Satellite" domain ("SimpleTime" variant) of the 3rd IPC. Number in brackets are action durations. The number before the action indicates its start time

median values over five runs for each problem considered. The CPU-time limit for each run was 5 minutes, after which the termination was forced.[8]

In order to verify the advantage of distinguishing different types of $\prec_C$ and $\prec_E$ constraints, we compared two versions of LPG: a version using TDA-graphs with all types of ordering constraints, and a simpler version in which every pair of actions can be ordered using only constraints of type $\overset{ES}{\prec}$ (both for $\prec_C$ and $\prec_E$ constraints). While this is a sound way of ordering actions, as discussed in the previous section, it can over-constrain the temporal order of two actions related by a causal or mutex relation. In the rest of the paper the simpler temporal plan representation will be denoted by TA-graph (instead of TDA-graph).

As test domains we used the version SimpleTime of Satellite, a domain used in the 3rd IPC. In this domain a "`turn_to`" action might be mutex with a "`calibrate`" action, because the first has the `at start` effect `(pointing ?s ?d_new)` denying an `at start` condition of the second. For instance, in the TDA-graph representation there is an $\overset{SS}{\prec}_E$*-constraint* between the `(calibrate satellite0 instr0 station2)` and `(turn_to satellite0 phen4 station2)` because the `at start` effect `(not (pointing satellite0 station2))` of the `turn_to` action denies the `at start` condition `(pointing satellite0 station2)` of the `calibrate` action. Therefore, in the TDA-graph representation the satellites can turn immediately after the beginning of the calibration of their instruments, while in the TA-graph representation the satellites can turn only at the end of the calibration, because all mutex actions are ordered by constraints of type $\overset{ES}{\prec}_E$. As a consequence, by using TDA-graphs LPG can find plans of better quality, i.e., of shorter makespan. Figure 5 shows a plan found by LPG containing the actions discussed above. Note that `(turn_to satellite0 phen4 station2)` starts at time 5.004, because of the use of $\overset{SS}{\prec}_E$ constraint between `calibrate` and `turn_to`. If we used a (stronger) $\prec_E$*-constraint*, `turn_to` would have to start after time 10.003, leading to a plan longer by five time units. Although the overlapping

[8] The tests of LPG were conducted on a PIII Intel 866 Mhz with 512 Mbytes of RAM. As described in [6] and [5], LPG is an incremental, anytime planner producing a succession of plans, each of which improves the quality of the previous plans.

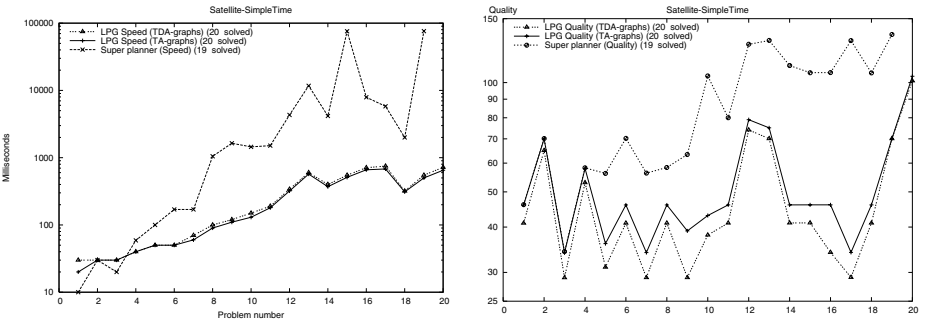



**Fig. 6.** Performance of TDA-graphs compared with TA-graphs and the SuperPlanner in Satellite, SimpleTime. On the x-axis we have the problem names indicated with numbers. On the y-axis (log scale), we have the CPU-time (left) and the quality of the plans measured using the makespan of the plan (right)

of the mutex actions previously illustrated is the only possible one in the Satellite domain, the duration of the plans generated by LPG using TDA-graphs is on average 10% shorter than the duration of the plans generated using TA-graphs. Moreover, in terms of CPU-time the overhead incurred by handling TDA-graphs instead of TA-graphs was negligible (see figure 6).

In order to derive some general results on the performance of our planner with respect to all the other planners of the 3rd IPC, we compared the results of the last version of our planner with the best results over all the other fully automated planners in terms of CPU-time and plan quality. We will indicate the second results as if they were produced by an hypothetical "SuperPlanner" (note, however, that such a planner does not exist). [9]

The performance of LPG was tested in terms of both CPU-time required to find a solution (LPG-speed) and quality of the best plan computed, using at most 5 minutes of CPU-time (LPG-quality). The overall results are showed in table 2. LPG-speed is generally faster than the SuperPlanner, and it always solves a larger number of problems. Overall, the percentage of the problems solved by LPG is 98.7%, while those solved by the SuperPlanner is 75%. The percentage of the problems in which our planner is faster is 87.5%, while this percentage for the SuperPlanner is 10.2%. Concerning LPG-quality, the percentage of the problems for which our planner produced a better quality solution is 94.6%, while this percentage for the SuperPlanner is only 5.4%. In particular LPG finds a solution with quality considerably better (at least 50%) in 35.7% of the problems for which both LPG and SuperPlanner find a solution (with some significant differences in Satellite), while the SuperPlanner never finds a solution with quality considerably better than LPG.

[9] The tests of SuperPlanner were conducted on the official machine of the competition, an AMD Athlon(tm) MP 1800+ (1500Mhz) with 1 Gbytes of RAM, which is slightly faster than the machine used to test the last version of LPG.

**Table 2.** Summary of the comparison of LPG and SuperPlanner in terms of: number of problems solved by LPG and the SuperPlanner (2nd/3rd columns); problems in which LPG-speed is faster/slower (4th/5th columns); problems in which LPG-speed is about one order of magnitude faster than SuperPlanner (6th column), problems in which LPG-quality computes better/worse solutions (7th/8th columns); problems in which LPG-quality computes much better solutions (9th column), defined as plans with duration at least 50% shorter than the duration of the plans produced by the SuperPlanner

| Domain | Problems solved by | | LPG-speed | | | LPG-quality | | |
|---|---|---|---|---|---|---|---|---|
| | LPG | SuperP. | better than SuperP. | worse than SuperP. | much better than SuperP. | better than SuperP. | worse than SuperP. | much better than SuperP. |
| **Simple-time** | | | | | | | | |
| Depots | 22 (100%) | 11 (50%) | 19 (86.4%) | 3 (13.6%) | 15 (68.2%) | 10 (90.9%) | 1 (9.1%) | 9 (81.8%) |
| DriverLog | 20 (100%) | 16 (80%) | 18 (90%) | 2 (10%) | 6 (30%) | 15 (93.8%) | 1 (6.2%) | 2 (12.5%) |
| Rovers | 20 (100%) | 10 (50%) | 16 (80%) | 4 (20%) | 12 (60%) | 10 (100%) | 0 (0%) | 0 (0%) |
| Satellite | 20 (100%) | 19 (95%) | 17 (85%) | 2 (10%) | 9 (45%) | 19 (100%) | 0 (0%) | 14 (73.7%) |
| ZenoTravel | 18 (90%) | 16 (80%) | 17 (85%) | 1 (5%) | 5 (25%) | 15 (93.8%) | 1 (6.2%) | 1 (6.2%) |
| Total | 98% | 70.6% | 85.2% | 11.7% | 46% | 95.8% | 4.2% | 36.1% |
| **Time** | | | | | | | | |
| Depots | 21 (95.5%) | 11 (50%) | 14 (63.6%) | 7 (31.8%) | 13 (59.1%) | 10 (90.9%) | 1 (9.1%) | 2 (18.2%) |
| DriverLog | 20 (100%) | 16 (80%) | 19 (95%) | 1 (5%) | 6 (30%) | 16 (100%) | 0 (0%) | 5 (31.2%) |
| Rovers | 20 (100%) | 12 (60%) | 19 (95%) | 1 (5%) | 13 (65%) | 12 (100%) | 0 (0%) | 0 (0%) |
| Satellite | 20 (100%) | 20 (100%) | 18 (90%) | 1 (5%) | 12 (60%) | 20 (100%) | 0 (0%) | 15 (75%) |
| ZenoTravel | 20 (100%) | 20 (100%) | 20 (100%) | 0 (0%) | 0 (0%) | 15 (75%) | 5 (25%) | 0 (0%) |
| Total | 99% | 77.5% | 88.2% | 9.8% | 43.1% | 92.4% | 7.6% | 27.8% |
| **Complex** | | | | | | | | |
| Satellite | 20 (100%) | 17 (85%) | 19 (95%) | 1 (5%) | 13 (65%) | 17 (100%) | 0 (0%) | 12 (70.6%) |
| Total | 98.7% | 75% | 87.5% | 10.2% | 46.4% | 94.6% | 5.4% | 35.7% |

## 5 Conclusions and Future Works

We have presented a new plan representation supporting durative actions, as well as some techniques for managing temporal information associated with facts and actions in a temporal plan. These techniques are fully implemented and integrated in the current version of LPG. As shown by the experimental results presented in this paper, the use of the TDA-graph representation in our planner supports the generation of very high quality plans, which in general are better than the plans computed by the other fully-automated planners that took part in the competition.

Our current temporal representation cannot deal with a particular case of action overlapping. This case can arise in domains where actions can be planned one during the other because of a particular combinations of different types of preconditions and effects. For instance, consider two actions $a$ and $b$ such that $dur(b) > dur(a)$. In principle, it can be possible that $a$ supports an `at end` condition of $b$ though an `at end` effect, and that $b$ supports an `at start` condition of $a$ though an `at start` effect. However, if $a$ has a precondition that is supported (only) by $b$, our planner cannot generate a plan in which $a$ is used to support a precondition of $b$. Future work includes an extension of TDA-graphs to handle these cases.

## References

[1] Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300. 91, 92, 95

[2] Fox, M., and Long, D. 2001. PDDL2.1: An extension to PDDL for expressing temporal planning domain. http://www.dur.ac.uk/d.p.long/competition.html. 91, 93

[3] Fox, M., and Long, D. 2003. The 3rd International Planning Competition: Results and Analysis. In *Journal Artificial Intelligence Research* (to appear). 92, 93, 97

[4] Gerevini, A., and Serina, I. 1999. Fast planning through greedy action graphs. In *Proceedings of AAAI-99*. 91, 92

[5] Gerevini, A., Serina, I., Saetti A., Spinoni S. 2003. Local Search for Temporal Planning in LPG. In *Proceedings of ICAPS-03*. 92, 93, 99, 101

[6] Gerevini, A., and Serina, I. 2002. LPG: A planner based on local search for planning graphs with action costs. In *Proceedings of AIPS-02*. 91, 92, 101

[7] Gerevini, A., Saetti, A., and Serina, I. 2003. Planning through Stochastic Local Search and Temporal Action Graphs. In *Journal Artificial Intelligence Research* (to appear). 93, 95, 98

[8] Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253–302. 91

# An Abductive Proof Procedure Handling Active Rules

Paolo Mancarella and Giacomo Terreni

Dipartimento di Informatica
University of Pisa
Italy
{paolo,terreni}@di.unipi.it

**Abstract.** We present a simple, though powerful extension of an abductive proof procedure proposed in the literature, the so-called KM-procedure, which allows one to properly treat more general forms of integrity constraints than those handled by the original procedure. These constraints are viewed as active rules, and their treatment allows the integration of a limited form of forward reasoning within the basic, backward reasoning framework upon which the KM-procedure is based. We first provide some background on Abductive Logic Programming and the KM-procedure and then formally present the extension, named AKM-procedure. The usefulness of the extension is shown by means of some simple examples.

## 1 Introduction and Motivations

In recent years, abduction and abductive reasoning have received a great attention in the AI community. It is now widely recognized that abduction provides a flexible and modular representation framework allowing a high-level representation of problems close to their natural specification. It has been shown to be a very useful representation and reasoning framework in many AI domains and applications, including, among others, diagnosis, planning, scheduling, natural-language understanding and learning. In particular, abduction has been widely studied in the context of logic programming, starting from the work of Eshghi and Kowalski on negation as failure (NAF) and abduction [3]. Abductive Logic Programming (ALP) [9, 6, 15, 7] is nowdays a popular extension of the logic programming paradigm which allows to perform abductive reasoning within the logic programming paradigm. One important aspect of ALP is that many proof procedures have been developed which provide a computational effective counterpart of the abstract theoretical model. Among others we mention the KM-procedure [9, 8], the SLDNFA-procedure [2], the IFF-procedure [4].

More recently, abductive logic programming has been shown to be a useful representation framework in the context of multi-agent systems. In particular, the work in [12] show that agents and agents beliefs can be suitably represented as abductive logic programs with integrity constraints. In this context, abductive proof procedures are exploited to provide the execution model for the reasoning

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 105–117, 2003.
© Springer-Verlag Berlin Heidelberg 2003

capabilities of the agents. However, some of the proof procedures are not powerful enough to provide the required reasoning capabilities, whereas some others, though being powerful enough, are not feasible from a practical, computational point of view.

In this paper we concentrate on one particular aspect of the reasoning capabilities that may be useful in the abductive logic programming representation of agents within multi-agent systems. As shown in [12], in the ALP representation of agents integrity constraints play a central role as far as the representation of dialogues and negotiation protocols is concerned. In many situations, these integrity constraints are condition-action like rules which, in the execution model, should be handled properly in order to perform some kind of *forward* reasoning [13]. Some of the proof procedures mentioned above are natural extensions of the basic computational paradigm of logic programming, which is based instead on *backward* reasoning. Hence, these procedures need to be extended in order to cope with this new form of integrity constraints.

Here, we show how the KM-procedure can be extended in order to handle integrity constraints which are more general than those handled in the original formulation given in [9]. These constraints are in the form of condition-action rules and their treatment in the proposed extension preserves the computational feasibility of the original KM-procedure. Let us show an example to better understand the problem. The following rules express the ability that a person has of buying some goods: a TV, (price: 500 Euros) or an air conditioner (1000 Euros)

$$buy(tv) \leftarrow pay(500)$$
$$buy(condit) \leftarrow pay(1000)$$

In an abductive LP framework, the *pay* predicate may be seen as an *abducible* predicate: atoms of the form $pay(x)$ can be assumed to build explanations for goals. For instance the explanation $\{pay(500)\}$ can be constructed as a solution for the goal $buy(tv)$. Similarly, the explanation $\{pay(500), pay(1000)\}$ can be constructed as an explanation for the conjunctive goal $buy(tv), buy(condit)$. Indeed, the KM-procedure computes such explanations for the given goals.
In many problems, abductive logic programs can be equipped with integrity constraints used to restrict the possible sets of explanations. E.g., to express that a person cannot pay, overall, 1500 Euros, we could add the integrity constraint[1]

$$\neg(pay(500), pay(1000)), \text{ written in the form } \leftarrow pay(500), pay(1000)$$

Now, the above abductive solution for the conjunctive goal $buy(tv)$, $buy(condit)$ would be discarded since it is inconsistent with the integrity constraint.

Assume now that we want to express the fact that a person can actually buy both goods, provided it asks for a loan. In the example at hand, assuming that the person has 1050 Euros, we could write a constraint of the form

$$pay(500), pay(1000) \rightarrow loan(450)$$

[1] Clearly, this example is oversimplified, and such a constraint can be formulated in a much more general form.

where *loan* is a new abducible predicate. Notice that in the last formula we have intentionally reversed the implication symbol, in order to highlight the fact that such a rule should be interpreted as a condition-action rule. In other words, the rule should not be used to perform *backward* reasoning, but instead it should be used in a *forward* manner in order to impose asking for a loan if the total expenditures are too high. Unfortunately, the KM-procedure does not provide a direct mechanism to handle such condition-action rule like integrity constraint. The constraint must be first transformed into a denial

$$\leftarrow pay(500), pay(1000), loan^*(450)$$

where $loan^*$ is a predicate symbol which is interpreted as the negation of the predicate *loan* (by adopting the view of negation as failure as abduction first proposed in [3]). This formulation of the integrity constraint gives rise to a counter-intuitive behavior of the KM-procedure. Indeed, if we run the goal $buy(condit)$ we obtain two possible explanations: the set $\{pay(1000), pay^*(500)\}$ and the set $\{pay(1000), loan(450)\}$. If we adopt the first explanation we cannot later decide to buy also the TV; on the other hand, with the second set we are obliged to ask for a loan, even if not strictly needed. Indeed, both explanations are obtained by enforcing the satisfaction of the integrity constraint once the hypothesis $pay(1000)$ has been made in order to make the original goal $buy(condit)$ succeed. The problem arises because the transformation of the original integrity constraint into its denial form looses the intuitive meaning of the integrity constraint itself, i.e. that the loan should be asked for only if we buy both goods.

The rest of the paper is organized as follows. In Section 2 we provide some background on abduction and Abductive Logic Programming, and we informally present the original KM-procedure. In Section 3 we formally present the proposed extension and we show its behavior on some simple, though meaningful, examples. Finally, in Section 4 we point out some promising directions for further improvements and extensions.

## 2 Abduction, ALP and the KM-procedure

In this section we give some background on abduction and on the KM-procedure within the context of ALP. We assume that the reader has some familiarity with the basic concepts and notations used in logic programming.

Abduction is a form of synthetic inference which allows one to draw explanations of observations. In its simplest form, from $\alpha \rightarrow \beta$ (a general rule) and $\beta$ (an observation) abduction allows one to assume $\alpha$ as a possible explanation of the observation. In the general case, this form of reasoning from observations and rules to explanations may lead to inconsistencies (as it happens, for instance, if $\neg\alpha$ holds). Hence, abductive explanations should be assumed only if they do not lead to inconsistencies.

In the context of logic programming, Eshghi and Kowalski first proposed in [3] an abductive interpretation of negation by failure which amounts to viewing negative literals in a normal logic program as a form of (abductive) hypotheses

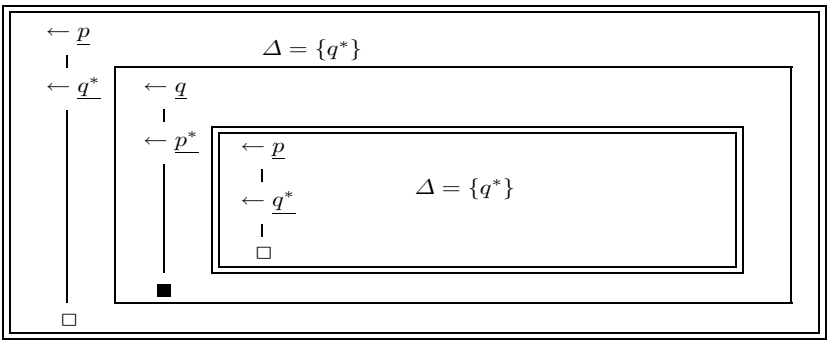



**Fig. 1.** A simple computation of the EK-procedure

that can be assumed to hold, provided they satisfy a canonical set of constraints which express the intended meaning of NAF.

From a semantics point of view, the abductive interpretation of NAF has a strong correspondence with the stable models semantics of normal logic programs [5]. From a computational viewpoint, Eshghi and Kowalski have extended the standard SLDNF proof procedure for NAF proposing an abductive proof procedure, referred to in the sequel as the EK-procedure, which terminates more often than the former, as shown in [11]. The computations of the EK-procedure are the interleaving of two types of phases: *abductive phases* in which possibly are generated hypotheses corresponding to the negative literals encountered during the computation, and *consistency phases* in which is checked that the generated hypotheses satisfy the constraints associated with NAF. In order to perform this checking, a consistency phase may in turn require new abductive phases to be fired. The following example shows a simple EK-procedure computation.

*Example 1.* Let $P$ be the following normal logic program.

$$p \leftarrow\sim q \qquad\qquad q \leftarrow\sim p$$

(where $\sim$ denotes NAF). It is clear that an SLDNF computation for the query $\leftarrow p$ will never terminate. Instead, the EK-procedure views negative atoms as hypotheses which can be assumed during a derivation, provided it is consistent to do so. The original normal logic program is first transformed into a positive program, by replacing NAF literals with positive atoms with a new predicate symbol. In the example, the transformation yields

$$p \leftarrow q^* \qquad\qquad q \leftarrow p^*$$

where $q^*$ and $p^*$ are new predicate symbols (representing the negation of $q$ and of $p$ respectively.) In the EK-procedure, the computation of $\leftarrow p$ succeeds by assuming the hypothesis $q^*$. The search space can be described as follows.

The part of the search space enclosed by a double box (resp. single box) corresponds to an abductive (resp. consistency) phase. The white little box represents success and the black one represents failure. A white box at the end of an

abductive (resp. consistency) phase corresponds to a success (resp. failure) of the phase, whereas a black box at the end of an abductive (resp. consistency) phase corresponds to a failure (resp. success) of the phase. Note that, in a consistency phase, the hypothesis to be checked is added to the current set $\Delta$ of hypotheses.

The abductive phase is basically SLD-resolution. When an abducible atom is selected (as $q^*$ in the outermost abductive phase of the search space), a consistency phase is fired in order to check for the consistency of the candidate hypothesis. In the case of the EK-procedure, since an hypothesis $q^*$ corresponds to the negation of $q$, checking its consistency amounts at making sure that $q$ fails. This is the aim of the consistency phase. As shown in the search space, the failure of $q$ amounts at ensuring that $p^*$ does not hold, and this is the reason why a nested abductive phase is fired, checking that $p$ is the case and hence it is safe *not* to assume $p^*$. Notice that the innermost abductive phase exploits the set of hypotheses collected so far. □

Kakas and Mancarella have then further extended this approach in order to handle also proper *abducibles* (beyond those used to model NAF). From a semantics point of view, they have defined an extension of stable models, namely *generalized* stable models, and from a computational point of view they have extended the EK-procedure into the so-called KM-procedure.

The KM-procedure is based on an initial abductive framework $\langle P, AB, IC\rangle$ where $P$ is a normal logic program, $AB$ is a set of predicate symbols called *abducibles* and $IC$ is a set of integrity constraints in denial form, i.e. in the form $\leftarrow L_1, \ldots, L_n$ where $L_1, \ldots, L_n$ are positive or negative literals. In the sequel, we will say that a literal $a(t)$ is *abducible* if $a \in AB$.

In the KM-procedure, negation is treated abductively as negation by failure as in the EK-procedure. Hence, the original abductive framework is first transformed into a new framework $\langle P^*, AB \cup AB^*, IC^* \cup I^*\rangle$, where: $P^*$ and $IC^*$ are obtained by substituting all the negative literals $\sim p(t)$ in $P$ and in $IC$ with NAF literals $p^*(t)$ (as described previously for the EK-procedure); $AB^*$ is the set of new predicate symbols $p^*$ for NAF and $I^*$ is the set of integrity constraints of the form $\leftarrow p(x), p^*(x)$, for each predicate symbol $p$.

To keep the notations as simple as possible, in the sequel we will not distinguish between the original framework and the transformed one. In other words we will write directly $\langle P, AB, IC\rangle$ meaning the framework obtained by the transformation that we have just described. In particular, we will not write explicitly the integrity constraints for NAF, and we will indicate in $AB$ only pure (positive) abducibles. In the same way, we will not show, in the examples, the computations related to NAF integrity constraints. Indeed, these computations are trivial.

Given a goal $G$, the KM-procedure computes a set of abducibles $\Delta \subseteq AB$ such that: (1) $P \cup \Delta \models G$ and (2) $P \cup \Delta$ satisfies $IC$.

Satisfaction of $IC$ should be interpreted as follows: for each ground instance $\leftarrow L_1, \ldots, L_n$ of a denial in $IC$, must hold that $P \cup \Delta \not\models L_1, \ldots, L_n$. If each integrity constraint contains at least one *pure* abducible, then the empty set of assumptions trivially satisfies $IC$. Indeed, let $a$ be a pure abducible and let $ic$

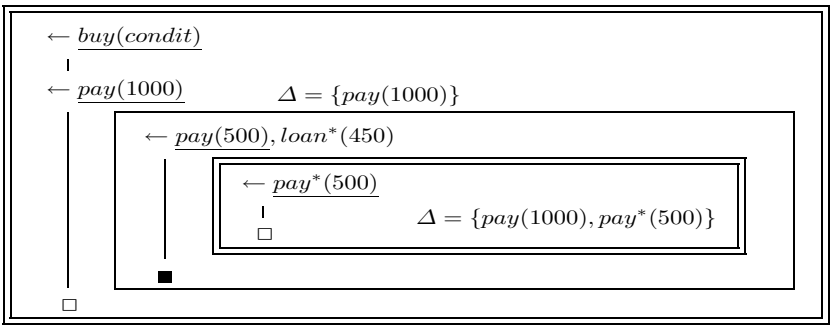



**Fig. 2.** A computation with the KM-procedure

be the ground instance $\leftarrow \ldots, a(t), \ldots$ of an integrity constraint. Given a set of abducibles $\Delta$, $P \cup \Delta$ may entail the conjunction $\ldots, a(t), \ldots$ only if $a(t) \in \Delta$.

The KM-procedure assumes that each integrity constraint satisfies the above condition, i.e. it contains at least one *pure* abducible. Then, the definition of the KM-procedure is a very simple extension of the EK-procedure. Again a computation is the interleaving of abductive derivations and consistency derivations. Since now *pure* abducibles may be selected during a derivation, their consistency must be checked against the integrity constraints in $IC$. This is a simple generalization of the consistency checking performed by the EK-procedure for the abducibles which are introduced for modelling NAF. We omit the details of the KM-procedure, since they are subsumed by the generalization we are going to give in the next section. Instead we show how the KM-procedure works on the example sketched in the introduction.

*Example 2.* Let us consider the following abductive framework.

$$\begin{array}{ll} P\colon buy(tv) \leftarrow pay(500) & AB = \{pay, loan\} \\ \quad buy(condit) \leftarrow pay(1000) & IC : \leftarrow pay(500), pay(1000), loan^*(450) \end{array}$$

Let us consider the goal $buy(condit)$. There are two possible successful computations using the KM-procedure. The first one yields the set of hypotheses (abducibles) $\Delta = \{pay(1000), pay^*(500)\}$, and the second yields the set $\Delta' = \{pay(1000), loan(450)\}$. We show the search space of the first computation.

The hypothesis $pay(1000)$, needed to succeed the outermost abductive derivation, fires the consistency checking of $\leftarrow pay(500), pay(1000), loan^*(450)$
This is done in the consistency phase, which amounts at ensuring that the residual of the integrity constraint, namely $\leftarrow pay(500), loan^*(450)$ fails. In this case, this is ensured by failing on the selected atom $pay(500)$ which in turn amounts at assuming $pay^*(500)$. The search space for the second computation is very similar: the only difference is that, in the consistency phase, the atom $loan^*(450)$ is selected, and its failure amounts at assuming $loan(450)$. □

The original semantics of the KM-procedure is based on the notion of generalized stable models [8], which is a two-valued semantics. The soundness of the procedure with respect to such semantics is guaranteed for a restricted class of programs (those which have at least one generalized stable model). This limitation has been solved by Toni in [15], where a three-valued argumentation based semantics [1] is given and it is shown that the KM-procedure is sound with respect to such semantics. Due to lack of space, we omit the definition of this semantics and refer the reader to [15] for the details.

## 3 KM-procedure with Active Constraints

In this section we propose an extension of the KM-procedure presented before, which is able to handle correctly a limited class of integrity constraints which represent condition-action like rules and which are named *active* constraints. This extension will be referred to as the AKM-procedure. The active constraints allow to integrate a limited, though powerful, form of forward reasoning within the basic abductive framework. Let be $A$ a literal and let be $L_1, \ldots, L_n$ *pure* abducible literals, active constraints are rules of the form $L_1 \wedge \ldots \wedge L_n \rightarrow A$.

The intuitive meaning of an active constraint as the one given above is that, whenever during a computation all the literals in the left hand side are contained in the current set of hypotheses, $A$ must be the case, i.e. it must be dynamically added to the current set of goals. Operationally, this amounts at adding $A$ to the current set of goals to be proven.

It is worth mentioning that such constraints can be handled in the KM-procedure, by transforming them into their denial form $\leftarrow A^*, L_1, \ldots, L_n$ Nevertheless, as we have already shown, this transformation has the undesired side-effect of loosing the intuitive "cause-effect" meaning of the original active rule.

In the sequel we will refer to an extended abductive framework $\langle P, AB, IC, AC \rangle$ where the newly added component $AC$ represents the set of active constraints.

The idea is to carry active constraints along with the computation, in order to partially evaluate their body with respect to the current set of hypotheses. So, at each step of the computation we will refer to the *current* set of active constraints as the set of partially evaluated active constraints. Initially the *current* set of active constraints coincides with $AC$ and whenever the body of an active constraint in the current set becomes empty, i.e. all the abducibles in it have been abduced, the head of the constraint is added to the current set of goals.

Assume that, at some step of the computation, the current set of active constraints is $AC_i$, the current set of hypotheses is $\Delta_i$ and a new abducible $\alpha$ is selected to be added to the current set of hypotheses. As usual, a consistency derivation checking the consistency of $\alpha$ must be fired. Before doing so, active constraints are analyzed to partially evaluate them with respect to the new set of hypotheses $\Delta_i \cup \alpha$. In what follows, we use the notation $Body \rightarrow A$ to denote a generic active constraint $\phi$ and we denote by $body(\phi)$ the set of literals in the

left hand side of such constraint. Moreover, given a literal $\alpha$, we write $\alpha^{not}$ to denote $a^*(t)$ if $\alpha = a(t)$, and $a(t)$ if $\alpha = a^*(t)$.

Informally, the new set of active constraints should be computed as follows.

(i) each constraint of the form $\phi$ such that $\alpha^{not} \in body(\phi)$ is deleted from the current set of active constraints;
(ii) each constraint of the form $Body \rightarrow \alpha$ is deleted from the current set of active constraints;
(iii) each active constraint $Body \rightarrow A$ such that $\alpha \in Body$ and $Body' = Body \setminus \{\alpha\}$ is not empty is replaced by $Body' \rightarrow A$;
(iv) each constraint of the form $\alpha \rightarrow A$ is deleted from the current set of constraints and $A$ is added to the current set of goals.

Let us give the formal specification of the AKM-procedure. As the KM-procedure, we define both abductive and consistency derivations as sequences of derivation steps. Before doing so, we need some auxiliary definitions and notations.

**Definition 1.** *Let $\Delta$ be a set of abducibles and $AC$ be a set of active constraints. By $\mathcal{A}(\Delta, AC)$ we denote the set $\{\phi \in AC \mid body(\phi) \cap \Delta \neq \{\}\ \}$* □

$\mathcal{A}(\Delta, AC)$ is the subset of $AC$ which can be partially evaluated w.r.t. $\Delta$. We then set up the definitions which formalize the steps (i) ÷ (iv) sketched above.

**Definition 2.** *Let $\alpha$ be an abducible and $AC$ be a set of active constraints. We define the sets $\mathcal{R}(\alpha, AC)$, $\mathcal{G}(\alpha, AC)$ and $\mathcal{T}(\alpha, AC)$ as follows:*

$$\begin{aligned}
\mathcal{R}(\alpha, AC) &= \{Body \rightarrow p \mid \phi \in \mathcal{A}(\{\alpha\}, AC) \wedge Body' = Body(\phi) \setminus \{\alpha\} \\
&\qquad \wedge\ Body' \neq \{\}\ \} \\
\mathcal{G}(\alpha, AC) &= \{p \mid (\alpha \rightarrow p) \in \mathcal{A}(\{\alpha\}, AC)\ \} \\
\mathcal{T}(\alpha, AC) &= \{Body \rightarrow \alpha \mid (Body \rightarrow \alpha) \in AC\ \}
\end{aligned}$$

□

$\mathcal{R}(\alpha, AC)$ is the result of partially evaluating the active constraints in $AC$ with respect to $\alpha$ as in case (iii) above. $\mathcal{G}(\alpha, AC)$ represents the set of goals which should be added to the current set of goals when $\alpha$ is abduced as in case (iv) above. Finally, $\mathcal{T}(\alpha, AC)$ represents the set of active constraints which should be get rid of as in case (ii) above. Notice that case (i) can be captured by the set $\mathcal{A}(\{\alpha^{not}\}, AC)$.

We are now in the position of presenting the AKM-procedure, by defining AKM-abductive derivations and AKM-consistency derivations. Abductive derivations are sequences of steps, each of which leads from a state $(G_i, \Delta_i, AC_i)$ to a state $(G_{i+1}, \Delta_{i+1}, AC_{i+1})$ where $G_i, G_{i+1}$ are sets of goals; $\Delta_i, \Delta_{i+1}$ are sets of abducibles and $AC_i, AC_{i+1}$ are sets of active constraints.

In a state $(G, \Delta, AC)$, $G$ is the current set of goals to be *achieved*, $\Delta$ is the set of abducibles assumed so far, and $AC$ is the current set of active constraints (dynamically obtained evaluating partially the original set w.r.t. $\Delta$).

On the other hand, consistency derivations are basically sequences of steps, each of which leads from a state $(F_i, \Delta_i, AC_i)$ to a state $(F_{i+1}, \Delta_{i+1}, AC_{i+1})$

where $F_i, F_{i+1}$ are sets of goals; $\Delta_i, \Delta_{i+1}$ are sets of abducibles and $AC_i, AC_{i+1}$ are sets of active constraints.

In a state $(F, \Delta, AC)$, $F$ is the current set of goals to be *failed*, $\Delta$ is the set of abducibles assumed so far, and $AC$ is the current set of active constraints.

At each (abductive or consistency) computation step, we assume that a *safe* computation rule is used. A computation rule is *safe* if, given a goal, it selects an abducible atom in it only if it is ground.

In what follows, we refer to an underlying abductive framework $\langle P, AB, IC, AC\rangle$.

**Abductive Derivation** Let $G_1$ be a set of goals, $\Delta_1$ be a set of abducibles and $AC_1$ be a set of active constraints. An abductive derivation from $(G_1, \Delta_1, AC_1)$ to $(G_n, \Delta_n, AC_n)$ is a sequence

$$(G_1, \Delta_1, AC_1), \ldots, (G_n, \Delta_n, AC_n) \text{ such that, } \forall i \in [1, \ldots, n]:$$

$G_i$ is of the form $S \cup G'_i$; $S = \leftarrow L_1, \ldots, L_k$ is the selected goal; $L_j$, $j \in [1, \ldots, k]$ is the atom selected by the comp. rule and $(G_{i+1}, \Delta_{i+1}, AC_{i+1})$ is obtained according to one of the following rules.

**Abd1** - $L_j$ is not abducible. Let $S'$ be the resolvent of some clause in $P$ with $S$ on $L_j$. Then:
- $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.
- if $S' \neq \Box$ then $G_{i+1} = S' \cup G'_i$; otherwise $G_{i+1} = G'_i$.

**Abd2** - $L_j$ is abducible and $L_j \in \Delta_i$. Let $S'$ be the resolvent of $S$ with $L_j$ (i.e. $S' = \leftarrow L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_k$). Then:
- $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.
- if $S' \neq \Box$ then $G_{i+1} = S' \cup G'_i$ otherwise $G_{i+1} = G'_i$.

**Abd3** - $L_j$ is abducible, $L_j \notin \Delta_i$ and $L_j^{not} \notin \Delta_i$. Then, let
$AC' = (AC_i \cup \mathcal{R}(L_j, AC_i)) \setminus (\mathcal{A}(\{L_j, L_j^{not}\}, AC_i) \cup \mathcal{T}(L_j, AC_i))$
$G'' = G'_i \cup \mathcal{G}(L_j, AC_i)$.
If there exists a consistency derivation from $(\{L_j\}, \Delta_i \cup \{L_j\}, AC')$ to $(\{\}, \Delta', AC'')$, then let $S'$ be the resolvent of $S$ with $L_j$ (i.e. $S' = \leftarrow L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_k$).
- $\Delta_{i+1} = \Delta'$, - $AC_{i+1} = AC''$.
- if $S' \neq \Box$, then $G_{i+1} = S' \cup G''$; otherwise $G_{i+1} = G''$.

Step **(Abd1)** is ordinary SLD-resolution. Step **(Abd2)** allows to reuse assumptions which have already been made, and again it can be seen as an ordinary SLD-resolution step using elements of $\Delta$ as fact rules. These two steps are identical to the corresponding steps in the KM-procedure. Step **(Abd3)** corresponds to the generation of new assumptions. The new abducible atom is added to the current set of hypotheses and its consistency is checked by means of a consistency derivation (see below). In this consistency derivation, the set of current active constraints is updated taking the new hypothesis into account. Note that, if a successful consistency derivation can be found, the abductive derivation is carried on by deleting the new assumption from the current goal and possibly by adding to the current set of goals the new goals fired by the active constraints.

**Consistency Derivation** Let $\alpha$ be an abducible atom, $\Delta_1$ be a set of abducibles and $AC_1$ be a set of active constraints. A consistency derivation from $(\alpha, \Delta_1, AC_1)$ to $(F_n, \Delta_n, AC_n)$ is a sequence
$(\alpha, \Delta_1, AC_1), (F_1, \Delta_1, AC_1), \ldots, (F_n, \Delta_n, AC_n)$ where:

1. $F_1$ is the set of all goals of the form $\leftarrow L_1, \ldots, L_k$ obtained by resolving the abducible $\alpha$ with the denials in $IC$, and $\Box \notin F_1$;
2. $\forall i \in [1, n]$: $F_i$ is a set of denials of the form $S \cup F'_i$; $S = \leftarrow L_1, \ldots, L_k$ is the selected denial; $L_j$, $j \in [1, \ldots, k]$ is the atom selected by the comp. rule and $(F_{i+1}, \Delta_{i+1}, AC_{i+1})$ is obtained according to one of the following rules.

**Con1** - $L_j$ is not abducible. Let $S'$ be the set of all resolvents of clauses in $P$ with $S$ on $L_j$. If $\Box \notin S'$, then:
- $F_{i+1} = S' \cup F'_i$; - $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.

**Con2** - $L_j$ is abducible and $L_j \in \Delta_i$. Then, let $S'$ be the resolvent of $S$ with $L_j$ (i.e. $S' = \leftarrow L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_k$). If $S' \neq \Box$, then
- $F_{i+1} = S' \cup F'_i$; - $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.

**Con3** - $L_j$ is abducible and $L_j^{not} \in \Delta_i$.
- $F_{i+1} = F'_i$; - $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.

**Con4** - $L_j$ is abducible, $L_j \notin \Delta_i$ and $L_j^{not} \notin \Delta_i$. Then:
1. if there exists an abductive derivation from $(\{\leftarrow L_j^{not}\}, \Delta_i, AC_i)$ to $(\{\}, \Delta', AC')$; then
   - $F_{i+1} = F'_i$; - $\Delta_{i+1} = \Delta'$; - $AC_{i+1} = AC'$.
2. otherwise, let $S'$ be the resolvent of $S$ with $L_j$ (i.e. $S' = \leftarrow L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_k$). If $S' \neq \Box$, then
   - $F_{i+1} = S' \cup F'_i$; - $\Delta_{i+1} = \Delta_i$; - $AC_{i+1} = AC_i$.

In a consistency derivation, the very first step from $(\alpha, \Delta_1, AC_1)$ to $(F_1, \Delta_1, AC_1)$ amounts at setting up the set $F_1$ of goals which should be shown to fail in order for the assumption $\alpha$ to be consistent. Step **(Con1)** and step **(Con2)** are SLD-resolution steps involving either a non abducible atom, or an abducible atom which has already been assumed. Notice that, in both cases, the resolution step is required not to produce the empty clause $\Box$. Indeed, if $\Box$ was generated, the corresponding branch of the search space would succeed and this would lead to a failure in the consistency derivation. In step **(Con3)** the selected branch of the search space is eliminated, due to the fact that an abducible is selected and its contrary belongs to the current set of assumption. This is enough to ensure the failure of the branch and hence the consistency of the corresponding path. Finally, in step **(Con4)** the failure of the branch on $L_j$ is ensured by looking for an abductive derivation of $L_j^{not}$ (case 1). If this derivation is found, then the selected branch is eliminated from the search space as in case **(Con3)**. Otherwise its failure is searched for on some other literal (case 2).

Let $\Delta$ be a set of assumptions and $AC$ be a set of active constraints. Then, a *successful* abductive derivation for a set of goals $G$ is an abductive derivation from $(G, \Delta, AC)$ leading to $(\{\}, \Delta', AC')$, for some $\Delta'$, $AC'$.

Similarly, a *successful* consistency derivation for an abducible $\alpha$ is a consistency derivation from $(\alpha, \Delta, AC)$ leading to $(\{\}, \Delta', AC')$ for some $\Delta'$, $AC'$.

Initial conditions: $\Delta = \{\}$ ; $AC = \{pay(500), pay(1000) \rightarrow loan(450)\}$

$\leftarrow \underline{buy(condit)}$

$\leftarrow \underline{pay(1000)}$ $\qquad \Delta' = \{pay(1000)\}$, $AC' = \{pay(500) \rightarrow loan(450)\}$

$\square$

**Fig. 3.** Computation of Example 2 with the AKM-procedure

Let us see the behavior of this extended procedure on an example related to Example 2.

*Example 3.* Let us consider the following extended framework.

$$P\colon buy(tv) \leftarrow pay(500) \qquad AB = \{pay, loan\}$$
$$buy(condit) \leftarrow pay(1000) \qquad AC : pay(500), pay(1000) \rightarrow loan(450)$$

Notice that, in Example 2 the constraint in $AC$ was transformed into the denial $\leftarrow pay(500), pay(1000), loan^*(450)$. Let us consider the goal $\leftarrow buy(condit)$. There is a successful abductive derivation from $(\{\leftarrow buy(condit), \{\}, AC)$ leading to $(\{\}, \{pay(1000)\}, AC')$ where $AC' = \{pay(500) \rightarrow loan(450)\}$. The search space corresponding to this derivation can be described as follows.

Notice that the assumption of $pay(1000)$ does not fire the active integrity constraint. This is kept in its partially evaluated form in the current set $AC'$. The solution provided by $\Delta'$ is the intuitive correct one: there is no need to abduce $pay^*(500)$ nor to abduce $loan(450)$ as done in the KM-procedure (see Ex. 2).

Consider now the initial conjunctive goal $buy(tv), buy(condit)$. There is a successful abductive derivation from $(\{\leftarrow buy(tv), buy(condit)\}, \{\}, AC)$ leading to $(\{\}, \{pay(1000, pay(500), loan(450))\}, \{\})$, with the following search space.

Initial conditions: $\Delta = \{\}$ ; $AC = \{pay(500), pay(1000) \rightarrow loan(450)\}$

$\leftarrow \underline{buy(tv)}, buy(condit)$

$\qquad \Delta_1 = \{pay(500)\}$; $AC_1 = \{pay(1000) \rightarrow loan(450)\}$

$\leftarrow \underline{pay(500)}, buy(condit)$

$\leftarrow \underline{buy(condit)}$

$\qquad \Delta_2 = \{pay(500), pay(1000)\}$; $AC_2 = \{\}$

$\leftarrow \underline{pay(1000)}$

$\leftarrow \underline{loan(450)}$

$\qquad \Delta_3 = \{pay(500), pay(1000), loan(450)\}$; $AC_3 = \{\}$

$\square$

**Fig. 4.** Use of active constraints with the AKM-procedure

The dashed line in the abductive derivation points out the fact that the new goal $loan(450)$ is not obtained through an SLD-derivation step, but it is inserted in the search space by the firing of the active integrity constraint. Indeed, when $pay(1000)$ is require to satisfy the original goal, the set $\Delta_2$ matches the body of the original active constraint, and this requires the firing of the new goal. □

In [14] the argumentation based semantics of [15] is extended in order to take active constraints into account. This semantics can be viewed as a further, three-valued extension of the generalized stable model semantics of [8], where the enforcement of active constraints is treated to model their intuitive behavior. In [14], the AKM-procedure is shown to be sound w.r.t. this extended semantics.

## 4 Conclusions and Future Works

The AKM-procedure allows the correct treatment of active constraints viewed as forward rules. This procedure overcomes some limitations of the KM-procedure due to the particular form of integrity constraints that the latter is able to handle. Our extension has the advantage of being simple and of being computationally feasible as the KM-procedure. Indeed, we are currently building a prototype implementation of the AKM-procedure, within the $ACLP$ framework [10].

There are several further extensions that are worth studying. The first extension is to consider active constraints in which the head is a conjunction. This can be easily incorporated into the AKM-procedure (indeed, we have limited our presentation to the case in which the head is a single atom only for keeping the presentation itself simple). More importantly, active constraints handled by the AKM-procedure are of a limited form, since their left hand side can contain only abducibles. Even though this form of constraints is adequate in many applications (such as the multi-agent applications mentioned in the introduction where (active) constraints are used to model negotiation protocols as in [12]), there may be situations in which more general forms of constraints are needed. One such form may involve allowing also non-abducibles to occur in the left hand side of the constraints. We could indeed relax this limitation by imposing that at least one abducible atom occurs in the left hand side of an active constraint. In this case, when during a computation, an active constraint is partially evaluated to $L_1, \ldots, L_n \rightarrow A$ and none of the remaining literals $L_i$s in the left hand side is abducible, the procedure should check whether or not the conjunction in the left hand side holds. If it does hold, then an abductive derivation for $A$ should be fired. Notice that checking that the conjunction $L_1, \ldots, L_n$ holds, should be done with respect to the *current* set $\Delta$ of hypotheses only. In other words, this checking should not be done through an abductive derivation, but instead through an ordinary SLD-derivation with respect to $P \cup \Delta$. Moreover, this checking should be repeated each time a new abducible is added to the current set of hypotheses.

## Acknowledgments

This work was done within the Information Society Technologies programme of the European Commission under the IST-2001-32530 project *SOCS*.

## References

[1] A. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence 93(1-2)*, pp. 63-101. 1997. 111

[2] M. Denecker and D. De Schreye. SLDNFA: an abductive procedure for normal abductive programs. In K. Apt, editor, *Proc. International Conference and Symposium on Logic Programming*, pp. 686–700, MIT press, 1992. 105

[3] K. Eshghi, R. A. Kowalski. Abduction compared with negation by failure. *Proc. 6th International Conference on Logic Programming*, pp. 234-254. MIT Press, 1989. 105, 107

[4] T. H Fung and R. A. Kowalski. The iff procedure for abductive logic programming. *Journal of logic programming 33(2)*, pp. 151-165. Elsevier, 1990. 105

[5] M. Gelfond and V. Lifschitz. The stable model semantics for logic programs. In K. Bowen and R. A. Kowalski, editors, *Proc. International Conference and Symposium on Logic Programming*, pp. 1070–1080, MIT press, 1988. 108

[6] A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1993. 105

[7] A. C. Kakas, R. A. Kowalski, F. Toni. The role of abduction in Logic Programming. *Handbook of Logic in AI and Logic Programming 5*, pp. 235-324. OUP, 1998. 105

[8] A. C. Kakas and P. Mancarella. Generalised stable models: a semantics for abduction. *Proceedings 9th European Conference on AI*, pp. 385-391. Pitman, 1990. 105, 111, 116

[9] A. C. Kakas, P. Mancarella. Abductive Logic Programming. *Proceedings NACLP Workshop on Non-Monotonic Reasoning and Logic Programming*. Austin, 1990. 105, 106

[10] A. C. Kakas, A. Michael, and C. Mourlas. Aclp: Abductive constraint logic programming. *Journal of Logic Programming*, 44(1-3):129–177, July-August 2000. 116

[11] P. Mancarella, D. Pedreschi, and S. Ruggieri. Negation as failure through abduction: reasoning about termination. *Computational Logic: Logic Programming and Beyond*, Springer-Verlag LNAI 2407, pp. 240-272, 2002. 108

[12] F. Sadri, F. Toni, and P. Torroni. An abductive logic programming architecture for negotiating agents. *Proceedings of the 8th European Conference on Logics in Artificial Intelligence* (JELIA'02), Springer-Verlag LNAI 2424, pp. 419-431, 2002. 105, 106, 116

[13] F. Sadri and F. Toni. Abduction with Negation As Failure for Active and Reactive Rules. Department of Computing, Imperial College, London. 106

[14] G. Terreni. Estensione di procedure abduttive al trattamento di vincoli attivi. (in Italian), Laurea Degree Thesis, Dipartimento di Informatica, Univ. di Pisa, 2002. 116

[15] F. Toni. Abductive logic programming. PhD Thesis, Department of Computing, Imperial College, London. 1995. 105, 111, 116

# BackPropagation through Cyclic Structures

M. Bianchini, M. Gori, L. Sarti, and F. Scarselli

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Siena
Via Roma, 56 — 53100 Siena (ITALY)
{monica,marco,sarti,franco}@ing.unisi.it

**Abstract.** Recursive neural networks are a powerful tool for processing structured data. According to the recursive learning paradigm, the information to be processed consists of directed positional acyclic graphs (DPAGs). In fact, recursive networks are fed following the partial order defined by the links of the graph. Unfortunately, the hypothesis of processing DPAGs is sometimes too restrictive, being the nature of some real–world problems intrinsically disordered and cyclic. In this paper, a methodology is proposed which allows us to map any cyclic directed graph into a "*recursive–equivalent*" tree. Therefore, the computational power of recursive networks is definitely established, also clarifying the underlying limitations of the model. The subgraph–isomorphism detection problem was used for testing the approach, showing very promising results.

## 1 Introduction

In several applications, the information which is relevant for solving problems is encoded into the relationships between some basic entities. The simplest dynamic data type is the *sequence*, which is a natural way of modeling time dependences. Nevertheless, there are domains, such as document processing or computational chemistry, in which the information involved is organized in more complex structures, like trees or graphs. Recently, a new connectionist model, called *recursive network* and tailored for dealing with structured information, was proposed [1]. According to the recursive paradigm, the information to be processed consists of directed positional acyclic graphs (DPAGs). In fact, recursive networks are fed following the partial order defined by the links of the graph. Unfortunately, also the hypothesis of processing DPAGs is sometimes too restrictive, being the nature of some real–world problems intrinsically disordered and cyclic. Examples of such problems are the classification of HTML pages [2], the image retrieval in multimedia databases [3, 4], and the prediction of the biological activity of chemical compounds [1]. In fact, Web pages can naturally be represented by graphs deduced directly by the HTML tags: nodes denote the logic contexts (paragraphs, sections, lists, etc.), while arcs denote the inclusion relationships between contexts and the connections established by the hyperlinks. The labels contain the words that are enclosed in the corresponding contexts. Such graphs

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 118–129, 2003.
© Springer-Verlag Berlin Heidelberg 2003

are typically ordered, directed, and cyclic. On the other hand, segmented images and complex chemical compounds can be coded as undirected cyclic graphs, whose nodes are labeled, respectively, by feature vectors and by atoms or simple molecules.
In this paper, the computational power of recursive neural networks is definitely established, also clarifying the underlying limitations. In fact, the main drawback of the recursive model consists in dealing with graphs where some nodes share the labels. Such a problem does not depend on the presence of cycles and arises also in acyclic graphs [5].

The paper is organized as follows. In the next section, some background topics on recursive neural networks are briefly reviewed. In Section 3, the main results proposed in this work are discussed, while Section 4 collects preliminary but promising results on the subgraph–isomorphism detection problem. Finally, some conclusions are drawn in Section 5.

## 2 Recursive Neural Network Models

The class of graphs that can be appropriately processed by recursive neural networks is that of directed positional acyclic graphs. Let $G = (V, E)$ be a DPAG, where $V$ is the set of nodes and $E$ represents the set of edges. In the classification and approximation settings, we shall require the DPAG $G$ either to be empty or to possess a *supersource*, i.e. a vertex $s \in V$ such that any other vertex of $G$ can be reached by a directed path starting from $s$. Such a requirement is strictly related to the processing scheme which will be subsequently described [1].
Given a graph $G$ and $v \in V$, $\mathrm{pa}[v]$ is the set of the *parents* of $v$, while $\mathrm{ch}[v]$ represents the set of its *children*. The *indegree* of $v$ is the cardinality of $\mathrm{pa}[v]$, while its *outdegree*, $\mathrm{od}[v]$, is the cardinality of $\mathrm{ch}[v]$. Each node is *labeled*, i.e. contains a set of domain variables, called *labels*. The presence of a branch $(v, w)$ in a labeled graph assesses the existence of some sort of causal link between the variables contained in $v$ and $w$.

*Recursive neural networks* are a generalization of recurrent networks particularly suited to learn graphs (see Fig. 1). Recursive networks have been already used in some applications [1]. In order to process a graph $G$, the recursive network is unfolded through the graph structure, producing the *encoding* network. At each node $v$ of the graph, the *state* $\mathbf{X}_v$ is computed by a feedforward network as a function of the input label $\mathbf{U}_v$ and the state of its children:

$$\mathbf{X}_v = f(\mathbf{X}_{\mathrm{ch}[v]}, \mathbf{U}_v, \theta_f), \tag{1}$$

with

$$\mathbf{X}_{\mathrm{ch}[v]} = [\mathbf{X}'_{\mathrm{ch}_1[v]}, \ldots, \mathbf{X}'_{\mathrm{ch}_o[v]}]', \quad o = \max_{v \in V}\{\mathrm{od}[v]\},$$

[1] Note that if a DPAG does not have a supersource, it is still possible to define a convention for adding an extra vertex $s$ with a minimal number of outgoing edges, such that $s$ is a supersource for the expanded DPAG [6].

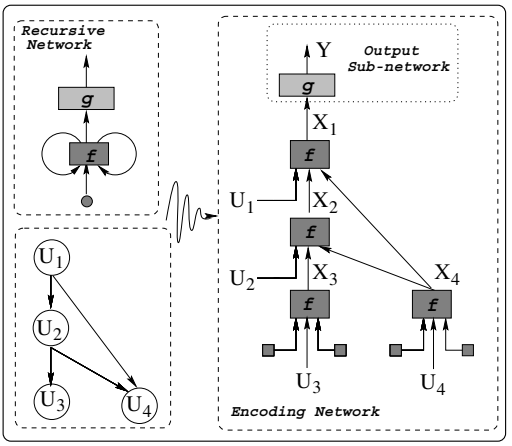



**Fig. 1.** The *encoding* and the *output* networks associated to a graph. The *recursive network* is unfolded through the structure of the graph

and $\mathbf{X}_{\mathrm{ch}_i[v]}$ equal to the *frontier state* $\mathbf{X}_0$, if node $v$ lacks of its $i$–th child.
At the supersource, also an output function is evaluated by a feedforward network, called the *output* network:

$$\mathbf{Y}_s = g(\mathbf{X}_s, \theta_g).$$

The parameters $\theta_f$ and $\theta_g$ are connection weights, being $\theta_f$ independent of node $v$ [2]. The parametric representations $f$ and $g$ can be implemented by a variety of neural network models. In the case of a two–layer perceptron, with sigmoidal activation functions in the hidden units and linear activation functions in the output units, the state is calculated according to:

$$\mathbf{X}_v = \mathbf{V} \cdot \boldsymbol{\sigma}\left(\sum_{k=1}^{o} \mathbf{A}_k \cdot \mathbf{X}_{\mathrm{ch}_k[v]} + \mathbf{B} \cdot \mathbf{U}_v + \mathbf{C}\right) + \mathbf{D}, \tag{2}$$

where $\boldsymbol{\sigma}$ is a vectorial sigmoidal function and $\theta_f$ collects $\mathbf{A}_k \in \mathbf{R}^{q,n}$, $k = 1, \ldots, o$, $\mathbf{B} \in \mathbf{R}^{q,m}$, $\mathbf{C} \in \mathbf{R}^q$, $\mathbf{D} \in \mathbf{R}^n$, and $\mathbf{V} \in \mathbf{R}^{n,q}$. Here, $m$ is the dimension of the label space, $n$ the dimension of the state space, and $q$ represents the number of hidden neurons. A similar equation holds for the output at the supersource:

$$\mathbf{Y}_s = \mathbf{W} \cdot \boldsymbol{\sigma}\left(\mathbf{E} \cdot \mathbf{X}_s + \mathbf{F}\right) + \mathbf{G},$$

where $\theta_g$ collects $\mathbf{E} \in \mathbf{R}^{q',n}$, $\mathbf{F} \in \mathbf{R}^{q'}$, $\mathbf{G} \in \mathbf{R}^r$, $\mathbf{W} \in \mathbf{R}^{r,q'}$.

**Remark:** In the case of sequential data (Fig. 2), recursive networks reduce to recurrent networks. In fact, the state updating described in eq. (2) becomes:

$$\mathbf{X}_t = \mathbf{V} \cdot \boldsymbol{\sigma}(\mathbf{A}\mathbf{X}_{t-1} + \mathbf{B}\mathbf{U}_t + \mathbf{C}) + \mathbf{D}.$$

[2] In this case, we say that the recursive neural network is *stationary*.

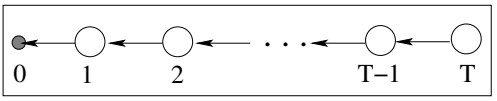



**Fig. 2.** A general sequence

Therefore, a recursive neural network implements a function $h$ : DPAGs $\rightarrow \mathbf{R}^r$, where $h(G) = \mathbf{Y}_s$. Formally, $h = g \circ \tilde{f}$, where $\tilde{f}(G) = \mathbf{X}_s$ denotes the process that takes a graph and returns the state at the supersource.
In [7] the ordering constraint is relaxed, so that the capability of recursive models to process directed acyclic graphs is assessed. On the other hand, in [8, 9], recursive neural networks are proved to be able to approximate, in probability, any function on ordered trees. More precisely, given a set $T \subset$ DPAGs of positional trees [3], a function $l : T \rightarrow \mathbf{R}^r$, a probability measure $P$ on $T$, and any real $\varepsilon$, there is a function $h$, realized by a recursive neural network, such that $P(|h(G) - l(G)| \geq \varepsilon) \leq \varepsilon$. The above result also characterizes the approximation capabilities of recursive neural networks w.r.t. the functions on DPAGs [5].

## 3 Recursive Processing of Cyclic Graphs

The recursive model just described cannot be directly applied to the processing of cyclic structures, because the unfolding of the recursive network would produce an infinite encoding network. In fact, eq. (1) gives rise to a sort of recursion in the definition of the states. A state $\mathbf{X}_v$ at a node $v$, involved in a cycle, is defined in terms of the same state $\mathbf{X}_v$, being $v$ a descendant of itself. In this way, the neural network works as a dynamical system, whose stable equilibrium points are the solutions of eq. (1).
In order to overcome the problem of cyclic structure processing, some researchers proposed to collapse each cycle in a unique unit, that resembles the whole information collected in the nodes belonging to the cycle [10]. Unfortunately, the collapse strategy cannot be carried out automatically and is intrinsically heuristic. Therefore, the effect on the resulting structures and the eventual loss of information are almost unpredictable.
In this paper, we propose a novel approach aimed at giving a new interpretation to eq. (1) in the case of cyclic graph processing. The graphs we consider are constrained to posses a supersource and to have distinct labels. According to our approach, the encoding network is a network with the same topology of the graph: if the graph is cyclic also the encoding network is cyclic (see Fig. 3). In fact, a copy of the transition network "replaces" each node of the graph and the connections between the transition networks are devised following the schema suggested by the arcs.

The computation is carried out by setting all the initial states $\mathbf{X}_v$ to $\mathbf{X}_0$ (see Algorithm 1.1). Then, the copies of the transition network are repeatedly

[3] A positional tree is a DPAG where each node has only one parent.

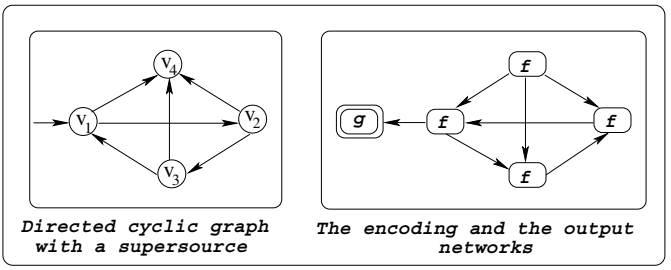



**Fig. 3.** The encoding and the output networks for a cyclic graph

activated to update the states. According to eq. (1), the transition network attached to node $v$ produces the new state $\mathbf{X}_v$ of $v$. After some updates, the computation can be stopped. The result of the output function can be regarded as the output of the whole recursive process. Our procedure is formalized in the following Algorithm 1.1.

**Algorithm 1.1 :** `CyclicRecursive(G)`
**begin**
**for each** $v \in V$ **do** $\mathbf{X}_v = \mathbf{X}_0$;
**repeat do**
  $<$ *Select* $v \in V >$;
  $\mathbf{X}_v = f(\mathbf{X}_{\mathrm{ch}[v]}, \mathbf{U}_v, \theta_f)$;
**until** stop();
**return** $g(\mathbf{X}_s, \theta_g)$;
**end**

Notice that no particular ordering is imposed on the sequence of activation of the transition networks. In fact, the transition networks can be activated following any ordering, and even random sequences are admitted. Moreover, both synchronous and asynchronous strategies can be adopted. Finally, how the stopping criterion for Algorithm 1.1 is realized is a fundamental issue and will be described in the next subsection.

In the following, we will prove that Algorithm 1.1 can effectively compute any function on cyclic directed graphs. Our rationale is based on the observation that recursive networks "behave" on cyclic directed graphs so as they process a particular class of recursive–equivalent trees. Once the recursive–equivalence relationship is established, theoretical results gained for function approximation on trees can be generalized to cyclic directed graphs. Therefore any function on graphs can be approximated, up to any degree of precision, by an *ad hoc* recursive network, where the transition function $f$ is appropriately chosen.

However, before proving the effectiveness of Algorithm 1.1 in approximating functions on cyclic structures, let us show a typical example in which the algorithm is applied to the processing of a complex image. The example will also constitute a practical demonstration of the biological plausibility of the proposed

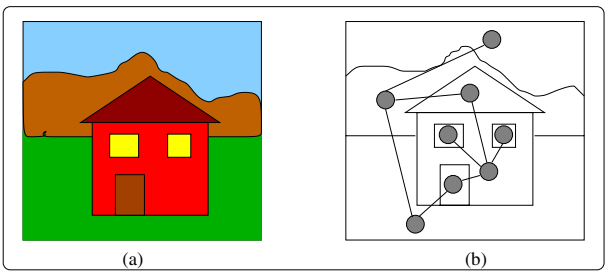



**Fig. 4.** An artificial image (a) and its RAG (b)

algorithm. In fact, the way in which it works resembles the reasoning of a person who analyzes a complex image by concentrating his/her attention on the components of the figure, wondering over all its pieces until he/she is able to produce a judgment.

**Example** In the last few years many efforts have been spent in order to devise engines to search images in large multimedia databases [4]. The established techniques are based on global or local perceptual features of the images, which are collected in a fixed–length array of reals. The retrieval task is based on a similarity criterion which is usually predefined by the particular choice of the feature vector.
Nevertheless, an image can be represented, in a more informative way, by its component regions and relationships among them [3]. In fact, once an image has been segmented [4], it is subdivided into $N$ regions, each described by a real valued array of features. On the other hand, the structural information associated to the relationships among regions can be represented by a graph. In particular, two connected regions $R_1$ and $R_2$ are *adjacent* if, for each pixel $a \in R_1$ and $b \in R_2$, there exists a path connecting $a$ and $b$, entirely lying into $R_1 \cup R_2$. In order to represent this type of structural information, the *Region Adjacency Graph* ($RAG$, see Fig. 4(b)) can be extracted from the segmented image by associating a node, labeled with the real feature vector, to each region, and linking the nodes associated to adjacent regions [5]. Then, it is possible to transform the RAG into a directed graph, by assuming that a couple of edges is attached to each undirected one, thus preserving the duplex information exchange.

When Algorithm 1.1 is applied to a RAG, the computation appears to follow an intuitive reasoning. At each time step, a region of the image is selected and the state of the corresponding node is computed, based on the states of some adjacent nodes. According to the recursive paradigm, the state of a node is an internal representation of the object denoted by that node. Thus, at each step, the algorithm adjusts the representation of a region using the representations of the adjacent regions. We will prove that there is a recursive network such

[4] An in–depth description on how the homogeneous regions of an image are extracted, using a segmentation algorithm, is out of the scope of this paper.
[5] In the RAG, the edges are not oriented.

that repeating those simple steps for a sufficient number of times allows us to compute any function [6] of the RAG. In fact, after the stopping criterion is satisfied, the state at the supersource (any node, in this case) should collect significant information on both the image perceptual features and the spatial arrangement.

### 3.1 Theoretical Results

In order to explain how Algorithm 1.1 can be effectively applied for processing cyclic structures, let us first describe how the set of cyclic directed graphs can be injectively mapped onto the set of trees.
Let $G = (V, E)$ be a cyclic directed graph with a supersource $s$, and with all distinct labels. The "*recursive–equivalent*" tree $T_o$ to a graph $G$ can be constructed as follows.

> A covering tree $T_c$ of graph $G$ is built, which is iteratively extended with other copies of nodes in $G$. The procedure halts when at least all the arcs of $G$ were visited. In fact, in the simplest case, a covering tree $T_c$ is extracted and extended with multiple copies of each node having an indegree larger than 1. Finally, in the output tree $T_o$, nodes with indegree $n$ appears $n$ times, whereas each arc in $E_o$ represents an arc in $E$. Otherwise, each arc and each node in $G$ should be visited many times, until the output at the supersource collects "sufficient" information on the cyclic structure.

Figs. 5(b) and 5(c) show examples of the two types of unfolding.

Anyway, given any function `CyclicGraphToTree` that implements the generic schema described above (see [11] for a formal algorithm), we will say that `CyclicGraphToTree`$(G)$ is *recursive–equivalent* to $G$. The following theorem proves that $G$ can be always directly reconstructed from each recursive–equivalent tree `CyclicGraphToTree`$(G)$.

**Theorem 1.** *Let $G = (V, E)$ be a cyclic directed graph with a supersource $s$, having all distinct labels. Let $T_o = (V_o, E_o)$ be a recursive–equivalent tree. $G$ can be uniquely reconstructed from $T_o$.*

*Proof.* The proof follows straightforwardly by noting that $G$ is obtained collapsing all the nodes with the same label belonging to $T_o$.

Therefore, any cyclic directed graph with distinct labels can be processed by a recursive neural network after the preprocessing phase described above. Thus, the results and the limitations derived upon tree–processing via recursive models can be directly extended to cyclic graphs.
Let $\mathcal{S}_{G_c}$ be the set of directed (eventually cyclic) graphs, having a supersource

[6] Examples of functions of the RAG are: deciding whether the figure is an house, computing the dimension of the largest region, deciding whether the image contains a black rectangle, etc..

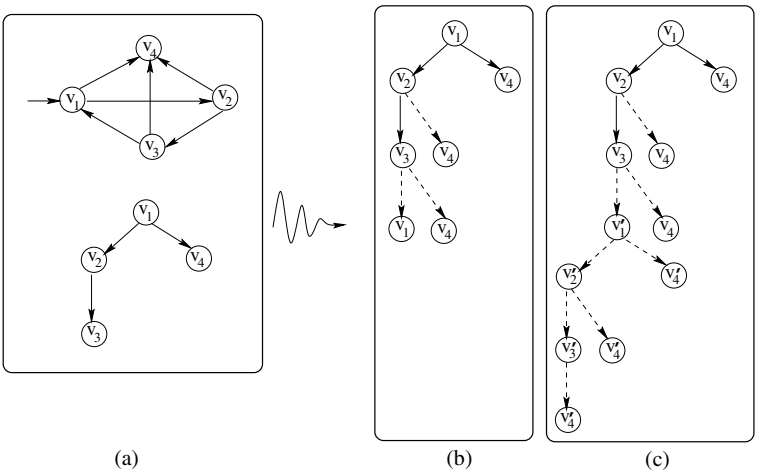



**Fig. 5.** A cyclic graph and a covering tree, in (a). Two recursive–equivalent trees, in (b) and (c)

and all distinct integer [7] labels. Any real function on $\mathcal{S}_{G_c}$ can be approximated in probability, up to any degree of precision, by a recursive neural network.

**Theorem 2.** *For any real function $l$, any probability measure $P$ on $\mathcal{S}_{G_c}$, and any real $\varepsilon$, there is a function $\mathcal{N}$, realized by a recursive neural network, such that $P(|l(G) - \mathcal{N}(\texttt{CyclicGraphToTree}(G))| \geq \varepsilon) \leq \varepsilon$.*

*Proof.* The proof follows straightforwardly from the results in [8].

In practice, the processing carried out by `CyclicGraphToTree` and the recursive neural network can be merged in a unique procedure, as Algorithm 1.2 shows.

**Algorithm 1.2** `:CyclicRecursProces(CurNode)`
**begin**
**if** `ContinueRecursion()`=**true then begin**
  <*Mark as visited all the arcs starting from* `CurNode`>;
  **for each** $v \in \mathrm{ch}[\texttt{CurNode}]$ **do**
    $\mathbf{X}_v = \texttt{CyclicRecursProces}(v)$;
  **return** $\mathbf{X}_{\texttt{CurNode}} = f(\mathbf{X}_{\mathrm{ch}[\texttt{CurNode}]}, \mathbf{U}_{\texttt{CurNode}}, \theta_f)$;
  **end**
**else return** $\mathbf{X}_{\texttt{CurNode}} = f(\mathbf{X}_0, \ldots, \mathbf{X}_0, \mathbf{U}_{\texttt{CurNode}}, \theta_f)$;
**end**

The procedure recursively applies eq. (1) starting from the supersource and visiting the graph. In Algorithm 1.2, `CurNode` denotes the node which is currently

[7] The result can also be extended to rational numbers provided that the considered labels are distinct up to a predefined $\delta > 0$.

visited. Since $\mathbf{X}_{\mathrm{ch}[\mathtt{CurNode}]}$ must be known in order to exploit eq. (1), the procedure `CyclicRecursProces` is applied to the children of the current node before computing $\mathbf{X}_{\mathtt{CurNode}}$. The recursion gives rise to a visit of the the graph (unfolding). The function `ContinueRecursion` controls the visit strategy. In fact, the recursion can be stopped at any time, the only constraint being that all the arcs of the graph must be covered at least one time. For nodes where the recursion is stopped, the state is computed assuming that the set of children is empty, otherwise the actual values of the states are used. Thus, the preprocessing procedure, which previously generated the recursive-equivalent tree, is now embedded in the tree processing.
Finally, we are able to prove that Algorithm 1.1 can compute any function on graphs.

**Theorem 3.** *For any real function $l$, any probability measure $P$ on $\mathcal{S}_{G_c}$, and any real $\varepsilon$, there is a recursive network $\mathcal{N}$, and a halt criterion stop, such that* $P(|l(G) - \mathtt{CyclicRecursive}(G)| \geq \varepsilon) \leq \varepsilon$.

*Proof.* See [11].

The main limitation of the recursive model consists in the difficulty of dealing with graphs where some nodes share the labels. Such a problem does not depend on the presence of cycles and arises also in acyclic graphs [5]. Nevertheless, the problem of shared labels may be overcame by some simple preprocessing. In fact, a graph having nodes with shared labels can be visited, automatically attaching to each label — which is generally represented by a record — a new field that contains a randomly generated integer. Many preprocessed structures can be obtained in this way from the same original graph. The recursive network should, therefore, be thought to recognize the equality of structures which differ only because of preprocessing.
Finally, notice that the network, whose existence is proved by Theorem 2, can be built by BackPropagation Through Structure [12], which is the common learning algorithm adopted for recursive networks, but the training set must be preprocessed in order to extract the set of recursive–equivalent trees. Since there are many equivalent trees for the same graph, it may be useful to transform each graph into a set of equivalent trees. In this way, if the training phase is successful, the network will produce the same output for every equivalent tree. Such a behavior of the recursive network implies that the output of Algorithm 1.2 becomes stable after some steps.

## 4 Experimental Results

The proposed method was evaluated on a significant task: the subgraph–isomorphism detection. In this kind of problem, a graph is explored to discover if it contains a given subgraph or not. The subgraph–isomorphism detection problem is interesting because it is often encountered in pattern recognition applications. Usually, the graph represents an object and the subgraph represents a pattern

**Table 1.** Training, Validation, and Test Sets

| | Training | Validation | Test |
|---|---|---|---|
| **Subgraph present** | 6968 | 1730 | 7062 |
| **Subgraph not present** | 7032 | 1770 | 6938 |
| **Cyclic graph** | 11056 | 2769 | 11170 |
| Minimum number of nodes | 5 | 5 | 5 |
| Maximum number of nodes | 23 | 21 | 22 |
| Average number of nodes | 10 | 11 | 10 |
| **Acyclic graph** | 2944 | 731 | 2830 |
| Minimum number of nodes | 5 | 5 | 5 |
| Maximum number of nodes | 16 | 16 | 17 |
| Average number of nodes | 10 | 10 | 11 |

that must be searched into the object. The detection of a particular object inside a given image, or the localization of a sequence inside a DNA, are examples of the subgraph–isomorphism detection problem.
The experimental data consisted of synthetic graphs which were generated by the following procedure:

1. For each graph, the procedure takes as input a range $[n_1,n_2]$, the minimum number of edges $|E|$, and the maximum outdegree allowed $o$;
2. $|V|$ nodes are generated, where $n_1 \leq |V| \leq n_2$;
3. Two random integers $v_1$, $v_2 \in [0,|V|-1]$ are generated, and the edge $(v_1, v_2)$ is inserted into the graph, provided that the edge is not a self–connection $(v_1 = v_2)$ and that node $v_1$ has not reached the maximum outdegree $o$. Step 3 is repeated until $|E|$ edges are created;
4. If a supersource node does not exist, it is added using the algorithm described in [6].

Three sets were generated using the above procedure: a learning set, a validation set, and a test set. The learning and the test sets were used to train the neural network and to measure its performance, respectively, while the validation set was exploited to implement a cross–validation strategy.
Each cyclic graph in the sets was transformed into a recursive–equivalent tree. With respect to the chosen criterion to stop unfolding (see Subsection 3.1), first of all, each edge was visited one time; subsequently, the graph was unfolded until a stochastic variable $x$ becomes true. The variable $x$ was true with probability 0.4 and false with probability 0.6. This halt criterion was chosen to guarantee both the recursive–equivalence of the generated trees and a bounded number of nodes.

The data sets randomly generated contain graphs having a minimum number of nodes, between five and seven, ten edges, and a maximum outdegree of three. Each label attached to the nodes contains a random real in [0,50] (shared labels are avoided). In order to produce positive and negative examples, a small

**Table 2.** Accuracy and rejection rate

| | Not Present | Present | Reject | Total | Accuracy (%) | Rejection (%) |
|---|---|---|---|---|---|---|
| Not Present | 5071 | 933 | 934 | 6938 | 73.09 | 13.46 |
| Present | 1089 | 4960 | 1013 | 7062 | 70.23 | 14.34 |

**Table 3.** Results classified by subgraphs dimension and by cyclic or acyclic topology

| | Subgraphs with 3 nodes | Subgraphs with 4 nodes | Subgraphs with 5 nodes | Cyclic subgraphs | Acyclic subgraphs |
|---|---|---|---|---|---|
| Accuracy (%) | 70.98 | 72.69 | 82.85 | 74.9 | 72.07 |
| Rejection (%) | 12.04 | 11.27 | 7.55 | 10.49 | 11.29 |

subgraph was inserted, in a random position, into a half of the data. Random noise with a uniform distribution in [-1,1] was added to the label of the nodes before the subgraph was inserted. The characteristics of the sets obtained after the preprocessing phase are reported in Table 1.
The transition network has a two–layer architecture and is composed of five hidden units and ten state units. The confusion table containing the obtained results is described in Table 2. A network output that belongs to [0,0.4] or [0.6,1] is classified as subgraph presence or absence, respectively, while an output belonging to [0.4,0.6] is not classified (rejected).

It is interesting to notice how the accuracy increases as the searched subgraph dimension grows (see the first three columns of Table 3). A similar behaviour is observed when we consider the results with respect to cyclic and acyclic subgraphs (the fourth and the fifth columns of Table 3). In fact, the preprocessing phase maps cyclic subgraphs to larger subtrees with respect to acyclic graphs.

## 5 Conclusions

In this paper, we have proposed a methodology to process cyclic graphs using recursive neural networks. A preprocessing phase is used in order to injectively map graphs into trees, so that theoretical results about recursive processing of trees can be directly extended to generic graphs. Moreover, an algorithm able to combine the preprocessing and the recursive computation phases is described. Therefore, the range of applicability and the computational power of recursive neural networks is definitely assessed. Promising preliminary results on the subgraph–isomorphism detection problem show the practical effectiveness of the proposed method.

## References


[1] P. Frasconi, M. Gori, and A. Sperduti, "A general framework for adaptive processing of data structures," *IEEE Transactions on Neural Networks*, vol. 9, pp. 768–786, September 1998. 118, 119

[2] M. Gori, M. Maggini, E. Martinelli, and F. Scarselli, "Learning user profiles in NAUTILUS," in *International Conference on Adaptive Hypermedia and Adaptive Web–based Systems*, (Trento (Italy)), August 2000. 118

[3] C. De Mauro, M. Diligenti, M. Gori, and M. Maggini, "Similarity learning for graph–based image representations," in *Proceedings of* $3^{rd}$ *IAPR–TC15 Workshop on Graph–based Representations in Pattern Recognition*, (Ischia (Naples)), pp. 250–259, May 23–25, 2001. 118, 123

[4] M. Gori, M. Maggini, and L. Sarti, "A recursive neural network model for processing direct acyclic graphs with labeled edges," in *International Joint Conference on Neural Networks*, (Portland (USA)), July 2003. 118, 123

[5] M. Bianchini, M. Gori, and F. Scarselli, "Theoretical properties of recursive networks with linear neurons," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 953–967, 2001. 119, 121, 126

[6] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," *IEEE Transactions on Neural Networks*, vol. 8, pp. 429–459, 1997. 119, 127

[7] M. Bianchini, M. Gori, and F. Scarselli, "Processing directed acyclic graphs with recursive neural networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1464–1470, 2001. 121

[8] B. Hammer, "Approximation capabilities of folding networks," in *ESANN '99*, (Bruges, (Belgium)), pp. 33–38, April 1999. 121, 125

[9] M. Bianchini, M. Gori, and F. Scarselli, "Recursive networks: An overview of theoretical results," in *Neural Nets — WIRN '99* (M. Marinaro and R. Tagliaferri, eds.), pp. 237–242, Vietri (Salerno, Italy): Springer, 1999. 121

[10] A. Bianucci, A. Micheli, A. Sperduti, and A. Starita, "Analysis of the internal representations developed by neural networks for structures applied to quantitative structure-activity relationship studies of benzodiazepines," *Journal of Chemical Information and Computer Sciences*, vol. 41, no. 1, pp. 202–218, 2001. 121

[11] M. Bianchini, M. Gori, and F. Scarselli, "Recursive processing of directed cyclic graphs," in *Proceedings of WCCI–IJCNN 2002*, (Honolulu, Hawaii), pp. 154–159, IEEE Press, 2002. 124, 126

[12] A. Küchler and C. Goller, "Inductive learning in symbolic domains using structure–driven recurrent neural networks," in *Advances in Artificial Intelligence* (G. Görz and S. Hölldobler, eds.), pp. 183–197, Berlin: Springer, 1996. 126

# A Neural Architecture for Segmentation and Modelling of Range Data

Roberto Pirrone and Antonio Chella

DINFO – University of Palermo
Viale delle Scienze 90128 Palermo, Italy
{chella,pirrone}@unipa.it

**Abstract.** A novel, two stage, neural architecture for the segmentation of range data and their modeling with undeformed superquadrics is presented. The system is composed by two distinct neural stages: a SOM is used to perform data segmentation, and, for each segment, a multilayer feed-forward network performs model estimation. The topology-preserving nature of the SOM algorithm makes this architecture suited to cluster data with respect to sudden curvature variations. The second stage is designed to model and compute the inside-outside function of an undeformed superquadric in whatever attitude, starting form the $(x, y, z)$ data triples. The network has been trained using backpropagation, and the weights arrangement, after training, represents a robust estimate of the superquadric parameters. The modelling network is compared also with a second implementation, which estimates separately the parameters of the 2D superellipses generating the 3D model. The whole architectural design is general, it can be extended to other geometric primitives for part-based object recognition, and performs faster than classical model fitting techniques. Detailed explanation of the theoretical approach, along with some experiments with real data are reported.

## 1 Introduction

The framework of the present work is the development of the vision system for an autonomous robot which is able to recognize, grasp, and manipulate the objects located in its operating environment. This step is a crucial part of the whole robot design: vision processes has to be at the same time fast, robust and accurate to guarantee the correct perception of the essential elements that are present in the operating environment. Moreover, the kind of images processed from the visual component of the robot, and the features that can be extracted from them, affect the other sensors equipment, the shape and, to some extent, the mission abilities of the robot itself.

Object recognition is one of the most intriguing visual processes to be modeled in a robot vision system. Really, it is not so clear if we recognize an object using some sort of 3D mental model encoding structural relations between its parts, or if we learn and store in our memory different views of the object itself, that in turn allow recognition using a global matching with our actual perception.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 130–141, 2003.
© Springer-Verlag Berlin Heidelberg 2003

These two ways of thinking produced an interesting debate, during the last twenty years, both in the psychological and in the computer vision communities, giving rise to two main group of theories. On one hand, there is a group of theories that are commonly referred to as "recognition by parts" which assume that human beings perform a sort of volumetric segmentation of the perceived image where each part is related to the others by structural relations such as *above()*, *larger()*, *side()* and so on [14]. The object shape is thus described with few flexible primitives that can be related to each other in several ways. The most famous system in this area is JIM and its successors proposed by Biederman and his colleagues [2] [8]. JIM uses "geons" as geometric primitives, but other systems have been proposed in the computer vision community that use superquadrics [18] [3].

On the other hand several view-based theories of object recognition have been developed following the key idea of a global match between the perceived image and some image-like "views" stored in our long term memory. Several vision systems have been proposed in literature, each with its own definition of the concept of view, depending on the particular theory to be validated. In general, one can say that a view is a vector containing the spatial coordinates of some image features. Such coordinates are expressed relative to a common reference point. The two main approaches to view matching are the global (*holistic*) image matching proposed by Poggio, Edelman and their colleagues [5] [4] and the theory of alignment, proposed by Ullman, where two-dimensional image features are geometrically aligned with a three-dimensional object model [23] [22].

In principle, both the approaches described above can be used in a robot vision system. Two-dimensional feature matching is a classical paradigm in robot vision where the main task is navigation and exploration of the environment. Even in the case of manipulators, view alignment can be a good solution for the robot visual servoing problem where the environment is totally controlled as in plants or factories. In the case of a robot that operates in a partially known environment, and which is aimed to perform recognition, grasping and manipulation tasks, a rigorous knowledge about the geometry and the spatial relations between objects in a true 3D model of the world is needed. Starting from the previous considerations the part-based recognition paradigm has been selected for the robot vision system, using superquadrics as geometric primitives for objects modelling. The focus of the present work is on a novel neural technique to perform fast segmentation and superquadric estimation from range data, in order to provide the robot with an object recognition system.

Several approaches to segmentation and modeling of range data have been proposed in literature: all of them are based on iterative procedures to fit the model to data [21] [24] [6]. In general, these approaches address also segmentation of complex objects [10] [19]. In this work, a neural architecture is presented which performs segmentation of a range data set and estimates an undeformed superquadric model for each retrieved part of the scene. The architecture consists in two neural networks: a Self-Organizing Map (SOM) [9] to perform data segmentation, and a multi-layer feed-forward network trained with backprop-

agation [20] which is devoted to model estimation. The SOM network is used to encode the data distribution with a low number of units which, in turn, are used for clustering. The feed-forward network for model estimation has been designed with a suitable topology and units' activation functions in order to compute the inside-outside superquadric function, starting from the range points. Connections between units, and units' activation functions are chosen to obtain a redundant coding of the superquadric parameters vector, starting from the weights arrangement in the trained network. In this work two different implementations of the modelling network are presented: the first one estimates directly the superquadric parameters using a single network, while the other makes use of a couple of networks each devoted to estimate parameters for one of the two generating superellipses.

The rest of the paper is arranged as follows. In section 2 a detailed description of the neural architecture will be provided, along with some theoretical remarks about superquadrics that are needed to illustrate the design choices. Section 3 will report the detail of the training experiments, and the comparison between the two modelling networks, using real range data. Finally, in section 4 conclusion will be drawn, and future work topics will be addressed.

## 2 Description of the Architecture

The architecture presented in this work is designed to segment a range data set, and to model each part using the inside-outside function of a generic undeformed superquadric. The neural approach to this task has several advantages. First of all, both the phases of the process are fast. The SOM training algorithm preserves the topology of the data set in the arrangement of its units, so it represents a straightforward approach to perform a quick subsampling of the range points. Segments emerge as simply connected regions, and their borders are located near concavities. A regards the model estimation network, the error function used in backpropagation training is more simpler than the classical error functions proposed in model fitting literature, so its first and second derivatives are easier to compute. Moreover, the neural approach ensures that computation is distributed across several simple units, thus allowing for fast performances. Finally, the neural approach is robust with respect to the sensor noise. Each best-matching unit in the SOM is representative of a neighborhood of range points, and its displacement averages all of them. Input data are smoothed also by the feed-forward network training process, because it runs on a great number of points, so the contribution of the outliers is neglected. Finally, its weights are, to some extent, a redundant coding of the superquadric parameters; in this way, some critical parameters like the form factors and the pose angles are computed solving an overdetermined system of equations. In what follows, the two steps of the process are illustrated in more detail.

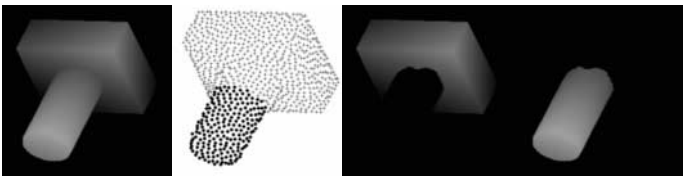


**Fig. 1.** From left to right: the HAMMER1 data set; clustering after SOM training and neural gas vector quantization; the resulting segments

### 2.1 Segmentation

The segmentation consists of a data subsampling process, clustering, and labeling phase. A SOM is trained on the whole data set, and the units in the map are arranged to follow the objects surfaces, due to the topology preserving property of this training algorithm. The units in the map tend to move towards the most convex surfaces of the data set, and are more sparse near concave regions. On the other side, neighborhood connections cause the activation of each unit to be influenced by closer ones, so the units tend to displace themselves as a sheet wrapping the data. Units' codebooks are then tuned using the well known neural gas algorithm [7]. In this way they are displaced exactly along the data surface, and a clear separation between them is obtained in those regions that correspond to occluding boundaries or concavities.

Clustering is performed using the k-means algorithm [12] with a variable number of clusters. In this phase, a measure of the global quantization error $e_{Qk}$ is computed for each run of the algorithm, and the right number of clusters is selected according to the rule:

$$c : e_{Qc} = \min_k (e_{Qk})$$

The quantization error is minimized by the same number of clusters as the convex blobs that are present in the data set because the neural gas trained SOM tends to group the units in these regions.

In the labeling phase each data point is assigned to the cluster that includes its best matching unit as results from the SOM training algorithm. The use of a SOM is needed not only to reduce the number of points to be clustered, but also to keep memory of the data points belonging to the sample neighborhood. In fig. 1 a segmentation example is reported for a real range data set.

### 2.2 Superquadrics

Superquadrics [1] are a powerful geometric primitive, that is widely used in computer vision to model real objects [17]. The model uses two form factors to undergo simple global variations from squared shapes to rounded, and pinched ones. Moreover, global deformation operators, like tapering or bending, have been defined, leading to a family of very expressive geometrical forms.

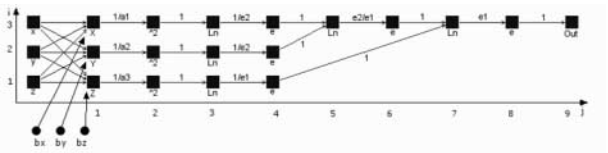



**Fig. 2.** Layout of the single network

In the case of an undeformed superquadric in a generic pose, a parameters vector made by 11 components is needed to model the shape: two form factors, three center coordinates, three axes sizes, and three pose angles (usually the Euler angles). The inside-outside equation of the primitive is defined by:

$$f(X,Y,Z) = \left[ \left( \frac{X}{a_1} \right)^{\frac{2}{\varepsilon_2}} + \left( \frac{Y}{a_2} \right)^{\frac{2}{\varepsilon_2}} \right]^{\frac{\varepsilon_2}{\varepsilon_1}} + \left( \frac{Z}{a_3} \right)^{\frac{2}{\varepsilon_1}} = 1 \tag{1}$$

where the generic point $\mathbf{X} = (X, Y, Z)$ is obtained by rotation and translation of the original data point $\mathbf{x} = (x, y, z)$, in order to refer it to the superquadric coordinate system. The direct transformation matrix to rotate and translate a point $\mathbf{X}$ expressed in the superquadric reference system to the point $\mathbf{x}$, in the world reference system, is:

$$\mathbf{x} = \mathbf{R}\mathbf{X} + \mathbf{t}$$

From the previous formula, the inverse transformation is:

$$\mathbf{X} = \mathbf{R}'\mathbf{x} + \mathbf{b} \ , \mathbf{b} \triangleq -\mathbf{R}'\mathbf{t} \tag{2}$$

### 2.3 Single Network for Model Estimation

Starting from equations (1) and (2) the layout of the network is reported in fig. 2. Here, the output of the network computes the modified equation $F(X,Y,Z) = f(X,Y,Z)^{\varepsilon_1}$. In this way we are guaranteed that the relation $F(\mathbf{p}) = k \ , \ k \neq 1$ implies that the geometrical fitting error in $\mathbf{p}$ is the same, regardless to the position of $\mathbf{p}$ . Input nodes represent the original point $\mathbf{x}$, while the first layer consists of full connected linear units which compute the $\mathbf{X}$ vector. The weights are the $r'_{ij}$ components of the rotation matrix reported in equation (2), while the three biases correspond to the components of the translation vector in the same equation. Apart from units in layer 2 whose activation is expressed by a square function, all hidden layers are made by interleaved logarithmic and exponential units. This choice derives form the need to have the form exponents as simple terms in a product, in order to treat them as weights of the units' connections. With this consideration in mind, each power in equation (1) has been expressed in the form: $a^x = \exp(x \ln(a))$. Following this approach, in the first $\ln - \exp$ couple of layers the powers of the $X$, $Y$, and $Z$ terms in equation (1) are computed, while the second power is performed on the sum of the $X$ and $Y$ terms. Finally,

the third couple performs rising all the inside-outside equation to the power of $\varepsilon_1$. In fig. 2 some weights are equal to 1: these connections have to remain fixed to be coherent with the model equation, so they will not be updated during the training process. It must be noted that this approach is applicable to whatever 3D primitive such as hyperquadrics, geons and so on, provided that it is possible to express the inside-outside function in a form where its parameters can be implemented as connections' weights between the units of a neural network.

The parameters vector is computed from the weights as follows. Considering the direct rotation matrix $\mathbf{R} = [R_{ij}]$, the approach proposed by Paul [16] can be adopted to derive the Euler angles $(\phi, \theta, \psi)$:

$$\begin{aligned} \phi &= \arctan\left(-\frac{R_{13}}{R_{23}}\right) \\ \theta &= \arctan\left(\frac{sin(\phi) - R_{23}\cos(\phi)}{R_{33}}\right) \\ \psi &= \arctan\left(-\frac{R_{22}sin(\phi) + R_{12}\cos(\phi)}{R_{21}sin(\phi) + R_{11}\cos(\phi)}\right) \end{aligned} \tag{3}$$

The vector of the center coordinates is derived from the biases values:

$$\mathbf{t} = -\mathbf{R}\mathbf{b} \tag{4}$$

The axes lengths can be obtained as:

$$a_k = \frac{1}{w_{1k,2k}}\ ,\ k = 1, 2, 3 \tag{5}$$

where $w_{ij,i+1k}$ is the weight of the connection between unit $j$ of the $i$-th layer and unit $k$ of the $i+1$-th layer, assuming that the input layer is labeled as layer 0. Using the previous notation, the form factors are computed from the following system of equations:

$$\begin{cases} w_{33,34}\varepsilon_2 & = 1 \\ w_{23,24}\varepsilon_2 & = 1 \\ w_{23,24}\varepsilon_1 & = 1 \\ w_{35,36}\varepsilon_1 - \varepsilon_2 & = 0 \\ w_{37,38} & = \varepsilon_1 \end{cases} \tag{6}$$

### 2.4 Double Network

Barr derives superquadrics in 3D space as a spherical product of two superconics defined in $\mathbb{R}^2$. We can think a superquadric expressed as in equation (1) as generated by two curves lying in the $(X, Y)$ and $(X, Z)$ plane respectively.

$$f(X, Y, Z) = f_1(X, Y) \otimes f_2(X, Z),$$

$$f_1(X, Y) = \left(\frac{X}{a_1}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{Y}{a_2}\right)^{\frac{2}{\varepsilon_2}},$$

$$f_2(X, Z) = \left(\frac{X}{a_1}\right)^{\frac{2}{\varepsilon_1}} + \left(\frac{Z}{a_3}\right)^{\frac{2}{\varepsilon_1}}$$

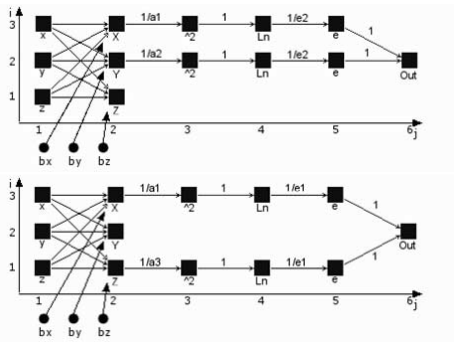



**Fig. 3.** The layouts for the double network

The double network for model estimation is arranged as a couple of feed-forward multi layer perceptrons, implementing the functions displayed above, and trained on two orthogonal tiny slices of data points, lying on the $(X, Y)$ and $(X, Z)$ plane in the canonical reference frame of the superquadric. The resulting layouts are depicted in fig. 3. Here, it is not necessary to perform rising of the whole function to the power of a form factor. Using the double network approach, we need only a small part of the data set to perform accurate estimation of the form factors, thus saving a lot of computation time. The estimation of the other parameters is performed by means of a modified version of the classical procedure introduced by Solina [21]. The steps of the numeric estimation for the parameters are reported in what follows.

1. The center of the model is computed as the center of mass of the contour points of each segment, not taking into account the rest of the data set.
2. Due to the model simmetry, all the data set is mirrored with respect to the center.
3. The maximum and minimum inertia directions are computed from the mirrored data set, and a first trial reference frame for the model is derived.
4. A guess iterative alignment procedure is performed, slightly rotating the reference frame, in order to maximize/minimize the moments of inertia computed in the step 3; in this way we reduce the effect of possible holes in the data produced by occlusions with other segments.
5. The direct rotation matrix $\mathbf{R}$ is computed from previous the estimate, using the Solina technique, while the Euler angles are computed as in equation (3).
6. The axes sizes are computed as the maximum distance of a point in the data set along the three coordinates of the reference frame.

In fig. 4 a reconstruction example is reported for the HAMMER1 data set, using both the single and the double network.

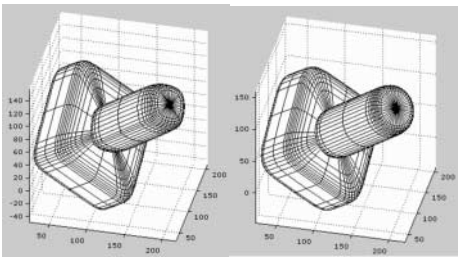


**Fig. 4.** The final reconstruction for the HAMMER1 data set, both for the single network (left) and the double one (right)

## 3 Experimental Setup

Experiments have been performed on a Pentium III 1GHz equipped with 256MB RAM, and running under MS Windows ME. The Stuttgart Neural Network Simulator (SNNS) v4.2 has been used to develop and train the modl estimation network. The $ln$ activation function, and the Levenberg-Marquardt learning function have been developed separately, using the SNNS API to compile them on the simulator kernel. The MATLAB SOM-Toolbox v.2.0 has been used to perform data segmentation.

Training samples have been selected from real range data available from the SEGMENTOR package developed by Solina and his colleagues [10].

In current implementation, the SOM software has been let free to determine the number $m$ of units, using the built-in heuristic formula $m = 5\sqrt{n}$, where $n$ is the number of data points. The units have been initialized linearly along the directions of the data eigenvectors, and the map sides have been computed from the ratio of the two main eigenvalues. The map has a hexagonal lattice, and has been trained using the batch version of the Kohonen algorithm, with a gaussian neighborhood function. Performances are very fast, despite the fact that the algorithm is run through the MATLAB kernel. For the data reported in fig. 1 the training took about 8 secs. for rough training and 16 secs. for fine training. Each trained SOM has been tuned using 50 epochs neural gas vector quantization. Finally, the k-means algorithm has been run varying the clusters number from 2 to 5 as there were no data sets with more than 4 connected regions.

Both the single, and the double network have been trained using backpropagation. The choice of this learning strategy is straightforward due to their multi-layer topology. Classical sum-of-squares (SSE) error function has been used to train the network because it is used in curve fitting problems, but its mathematical form is much simpler with respect to the metrics proposed in superquadrics fitting literature. Weights update has been performed both with the Levenberg-Marquardt (LeMa) algorithm [11] [13], the Scaled Conjugate Gradient (SCG) [15] approach, and some typical backpropagation schemes. In the case of the single network, the first two methods have proved to be faster and more ef-

**Table 1.** SSE values for the LeMa and SCG learning approaches, varying the training epochs

| *data set: HAMMER1* | **LeMa - Epochs** | | | | |
|---|---|---|---|---|---|
| *Seg. #* | *5* | *25* | *50* | *75* | *100* |
| *1* | 70.123 | 41.115 | 25.031 | 8.312 | 1.038 |
| *2* | 63.112 | 39.731 | 28.778 | 9.751 | 1.531 |
| | **SCG - Epochs** | | | | |
| *Seg. #* | *5* | *25* | *50* | *75* | *100* |
| *1* | 71.001 | 42.001 | 28.312 | 9.123 | 1.731 |
| *2* | 69.007 | 40.812 | 27.531 | 8.391 | 1.248 |

ficient than the classical gradient descent algorithm, in fact they are more suited to face the estimation of a high dimension non-linear model like superquadrics.

The network has been trained on all the segments of each data set, varying the number of learning epochs, and measuring the SSE value, in order to compare performances of the two learning functions. After some trials the learning rate has been fixed to 0.5 for all the experiments. The weights have been initialized using the approach proposed by Solina [21] to perform the initial estimation of the superquadric parameters. Using the inverse procedure of the one reported in eqs (3),(4),(5),(6) it is possible to obtain the values of the weights. In table 1 are reported the SSE values obtained for each segment, varying the number of learning epochs. Table 1 clearly shows that the two learning strategies have almost the same performance. This is a not surprising result, due to the simple mathematical form of the sum-of-squares error function. Moreover, one may argue that this result derives from an analogous finding, obtained when fitting the model to range data with classical error metrics. Besides, at least 100 learning epochs are needed to accomplish the training phase.

In the case of the double network, we found that the resilient backpropagation scheme resulted to be the faster approach, as it is shown in table 2 where resilient backprop and the SCG learning algorithm are compared. Using the resiliaent backpropagation it is possible to reduce the average learning epochs to almost 50. The whole time performance of the modelling architecture ranges from 2 to 5 secs. for the numerical preprocessing, while the learning takes a time between 0.5 and 4.5 secs. in the case of the single network, and between 0.3 and 0.7 secs. in the case of the double one. In fig. 5, some results are reported for the BENT (topmost row) and GEARBOX (bottom row) data sets.

## 4 Conclusions

A neural architecture aimed to segmentation and modeling of range data has been presented. The architecture consists of a SOM trained on data points, whose units are tuned using the neural gas algorithm, and clustered in convex sets with the k-means approach. Next a multi-layer feed-forward neural architecture is used to model the mathematical form of a superquadric's inside-outside

**Table 2.** SSE values for the SCG and resilient backpropagation learning approaches, varying the training epochs, in the double network

| *data set: HAMMER1* | **SCG - Epochs** | | | | |
|---|---|---|---|---|---|
| *Seg. #/net #* | *5* | *25* | *50* | *75* | *100* |
| *1/1* | 71.961 | 53.025 | 53.025 | 53.025 | 53.025 |
| *1/2* | 55.688 | 8.207 | * | * | * |
| *2/1* | 9.169 | 4.308 | 4.247 | 4.247 | 4.247 |
| *2/2* | 1.273 | * | * | * | * |
| | **Resilient Backprop - Epochs** | | | | |
| *Seg. #/net #* | *5* | *25* | *50* | *75* | *100* |
| *1/1* | 96.673 | 62.458 | 21.829 | * | * |
| *1/2* | 95.667 | 51.387 | 46.587 | 2.696 | * |
| *2/1* | 15.006 | 5.1445 | * | * | * |
| *2/2* | 1.368 | 1.031 | * | * | * |

function, taking also into account a generic spatial pose of the model. Two implementations have been compared: the first one estimates directly all the parameters, while the second uses a couple of network to estimate the generating superellipses. Model estimation is obtained training the networks on a set of range data, and the weights arrangement provides the estimation of the superquadric's parameters.

The proposed approach exhibits a satisfactory performance both as regards fastness, and robustness with respect to input noise. As a result of the SOM training the units are very close to the data set, while the neural gas algorithm clusters them away from concavities and occluding contours that are the boundaries of each convex data blob. Fast performances of the model estimation networks are due to the simplicity of the error function, and to the computation distribution across the network units. A good perceptual shape estimation has been obtained with all the experimented data sets. In particular, the double network performs a bit accurately than the single one, and is much faster.

Such a system is being implemented as a vision tool for a manipulator robot, in a visual servoing application, where an antropomorphic arm-hand system learns the motion of the operator, acquired by a stereo pair. The vision system enables the robot to create an internal (symbolic) representation of the spatial structure of the objects in its operating environment. This result can be achieved, despite the possible reconstruction errors, by means of a focus-of-attention mechanism that finds, at first, coarse regions of interest inside the input image. In the case of fine and precise movements, the process can be iterated with higher resolution only in those regions where the cognitive module of the robot architecture will focus its attention, thus saving a large amount of computational time.

Future work is oriented towards dynamic scenes. In such a framework, the static reconstruction approach explained so far, can be regarded as a starting point to create the internal representation of the scene. The robot can use this representation with a predictive tool, like the Kalman filter or the ARSOM net-

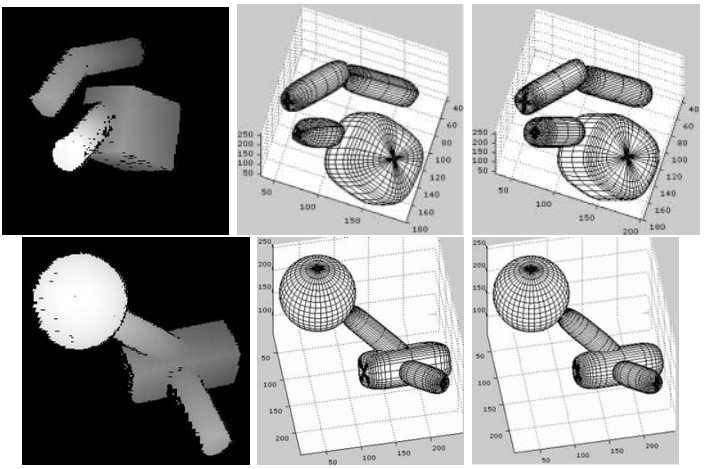


**Fig. 5.** From left to right: data set, single net modelling, double net modelling

work, to follow the scene dynamics (i.e. the evolution of the models parameters) without performing fine reconstruction, until predictions disagree with perception in a significant way.

## Acknowledgements

The authors want to thank Giampaolo Russo for the support in the implementation work that was part of his graduation thesis.

## References

[1] A.H. Barr. Superquadrics and Angle-preserving Transformations. *IEEE Computer Graphics and Applications*, 1:11–23, 1981. 133

[2] I. Biederman. Recognition-by-components: A theory of human image understanding. *Psychological Review*, 94(2):115–147, 1987. 131

[3] S.J. Dickinson, A.P. Pentland, and A. Rosenfeld. 3-D Shape Recovery Using Distributed Aspect Matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(2):174–198, 1992. 131

[4] S. Edelman. Representation is representation of similarities. *Behavioral & Brain Sciences*, 21:449–498, 1998. 131

[5] S. Edelman and T. Poggio. Bringing the grandmother back into the picture: A memory-based view of object recognition. A.I. Memo 1181, MIT, 1991. 131

[6] F.P. Ferrie, J. Lagarde, and P. Whaite. Darboux Frames, Snakes, and Super-Quadrics: Geometry From the Bottom Up. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(8):771–784, 1993. 131

[7] B. Fritzke. Growing Cell Structures — A Self-Organizing Network for Unsupervised and Supervised Learning. *Neural Networks*, 7(9):1441–1460, 1994. 133

[8] J.E. Hummel and I. Biederman. Dynamic binding in a neural network for shape recognition. *Psychological Review*, 99:480–517, 1992. 131

[9] T. Kohonen. The Self–Organizing Map. *Proceedings of the IEEE*, 78(9):1464–1480, September 1990. 131

[10] A. Leonardis, A. Jaklic, and F. Solina. Superquadrics for Segmenting and Modeling Range Data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(11):1289–1295, 1997. 131, 137

[11] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, II(2):164–168, 1944. 137

[12] J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman, editors, *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, CA, 1967. University of California Press. 133

[13] D.W. Marquardt. An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, 11(2):431–441, 1963. 137

[14] D. Marr. *Vision*. W.H. Freeman & Co, 1982. 131

[15] M. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993. 137

[16] R. Paul. *Robot Manipulators*. MIT Press, Cambridge, MA, 1981. 135

[17] A.P. Pentland. Perceptual organization and the representation of natural forms. *Artificial Intelligence*, 28:293–331, 1986. 133

[18] A.P. Pentland. Recognition by Parts. In *Proc. of International Conference on Computer Vision*, pages 612–620, London, 1987. 131

[19] R. Pirrone. Part based Segmentation and Modeling of Range Data by Moving Target. *Journal of Intelligent Systems*, 11(4):217–247, 2001. 131

[20] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning ingternal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1: Foundations, pages 318–362. MIT Press, Cambridge, MA, 1986. 132

[21] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(2):131–147, 1990. 131, 136, 138

[22] S. Ullman. *High-level Vision: Object Recognition and Visual Cognition*. MIT Press, Cambridge, MA, 1996. 131

[23] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13:992–1006, 1991. 131

[24] P. Whaite and F.P. Ferrie. From Uncertainty to Visual Exploration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(10):1038–1049, 1991. 131

# A Combination of Support Vector Machines and Bidirectional Recurrent Neural Networks for Protein Secondary Structure Prediction

Alessio Ceroni, Paolo Frasconi, Andrea Passerini, and Alessandro Vullo

Machine Learning and Neural Networks Group
Dipartimento di Sistemi e Informatica
Universitá di Firenze, Italy
Phone: +39 055 4796 361
Fax: +39 055 4796 363
http://www.dsi.unifi.it/neural/
{aceroni,paolo,passerini,vullo}@dsi.unifi.it

**Abstract.** Predicting the secondary structure of a protein is a main topic in bioinformatics. A reliable predictor is needed by threading methods to improve the prediction of tertiary structure. Moreover, the predicted secondary structure content of a protein can be used to assign the protein to a specific folding class and thus estimate its function.
We discuss here the use of support vector machines (SVMs) for the prediction of secondary structure. We show the results of a comparative experiment with a previously presented work. We measure the performances of SVMs on a significant non-redundant set of proteins. We present for the first time a direct comparison between SVMs and feed forward neural netwoks (NNs) for the task of secondary structure prediction. We exploit the use of bidirectional recurrent neural networks (BRNNs) as a filtering method to refine the predictions of the SVM classifier. Finally, we introduce a simple but effective idea to enforce constraints into secondary structure prediction based on finite-state automata (FSA) and Viterbi algorithm.

## 1 Introduction

Proteins are polypeptide chains carrying out most of the basic functions of life at the molecular level. These linear chains fold in complex 3D structures whose shape is responsible of proteins' behavior. Each ring of the chain consists of one of the 20 amino acids existing in nature. Therefore, a single protein can be represented as a sequence of letters from a 20 elements alphabet called the *primary structure* of the protein. The sequence of amino acids contains all the informations needed for a correct folding of the protein. Proteins are synthesized inside cells, using the instructions written in the DNA. The DNA contains genes, sequences of nucleotides *codifying* for proteins. Each triplet of nucleotides in a gene correspond to an amino acid of the encoded protein: the 64 possible configurations of three nucleotides encode the 20 symbols alphabet of amino

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 142–153, 2003.
© Springer-Verlag Berlin Heidelberg 2003

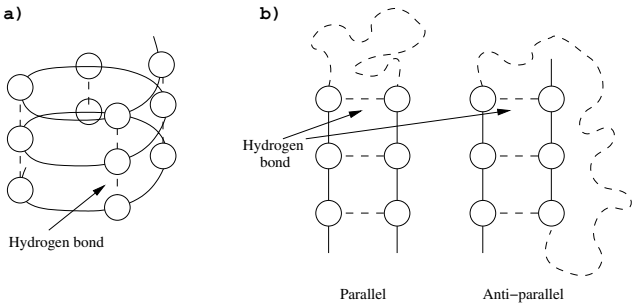



**Fig. 1.** Conformation of a) alpha helix and b) beta strands along the chain of a protein

acids (it's a redundant code), plus two symbols to identify start and end of the coding sequence.

All the observed proteins present local regularities in their 3D structure formed and maintained by hydrogen bonds between atoms. These regular structures are referred to as the protein's *secondary structure*. The most common configurations observed in proteins are called *alpha helices* and *beta strands*, while all the other conformations are usually referred to as *coils*. They are traditionally identified using a single letter code: $H$ (alpha helix), $E$ (beta strand) and $C$ (coil). An alpha helix is a cork screw like 3D structure formed by hydrogen bonded amino acids spaced three positions along the sequence (Figure 1a). On the contrary, a beta strand is a straight conformation of amino acids hydrogen bonded to the components of another strand, forming a planar aggregation called *beta sheet* (Figure 1b). A group of adjacent amino acids sharing the same conformation are members of a *segment* of secondary structure. Segments of secondary structure are well defined and stable aggregations of amino acids which strongly influence the chain's folding and which usually carry out specifical functions inside the protein, like words of a particular language forming a meaningful phrase.

Thanks to several genome sequencing projects, the entire DNA sequence of many organisms has been experimentally determined. Inside each genome the positions of genes have been discovered and the primary sequences of the corresponding proteins have been identified. Unfortunately, the proteins' 3D (*tertiary*) structure, essential to study their functions, remains almost unknown. Even if the number of proteins whose primary sequence is known counts in the number of millions, only few thousands of them have been succesfully crystallized and their 3D structure deposited in the Protein Data Bank [1]. It is therefore becoming increasingly important to predict protein's tertiary structure *ab initio* from its amino acid sequence. In this scenario a reliable secondary structure predictor plays a fundamental role. Most of the unknown tertiary structures can be inferred by *comparative modeling*, given the fact that proteins having similar primary structures tend to fold in similar ways. The remaining proteins are usually

predicted using *threading* algorithms, combining small pieces of other proteins that share local substructures with the query chain. However, these building blocks can be found only by performing a structure comparison, using an estimation of the protein secondary structure. Moreove, the predicted secondary structure content of a protein can be used to identify its folding family [2, 3] and thus estimate its functions.

The first attempt to apply machine learning techniques to the prediction of secondary structure [4] employed a standard multi layer perceptron (MLP) with a single hidden layer, and used as inputs a window of amino acids in one-hot code. The accuracy of this method, measured as the proportion of amino acids correctly assigned to one of the three secondary structure classes (three-state accuracy or $Q_3$) was well below 70%. The introduction of evolutionary information expressed by *multiple alignment* and *profiles*, represented a major contribution to the solution of the problem and allowed a significant improvement of the reported accuracy to about 72% [5]. A multiple alignment is a collection of sequences of amino acids from different proteins, realized using a maximum local alignment algorithm applied to a large database of known primary structures [6]. Once the multiple alignment has been computed, the profile is obtained by counting the frequency of each amino acid at every position in sequence, and used instead of the one-hot code as a representation of each amino acid.

A major drawback of using a MLP on a window of profiles is given by the relative independence between the predictions of adjacent positions in the sequence. On the contrary, the secondary structure of a protein is defined as a collection of segments composed by many consecutive amino acids. To quantify the capability of a classifier to correctly predict entire segments of secondary structure a measure of Segment OVerlap (SOV) is used [7]. A common approach used to improve both SOV and $Q_3$ is to employ a structure-to-structure classifier to filter the predictions of the first classifier. Jones [8] used neural networks (NNs) for both stages: thanks to this solution and to an increasing availability of training data, this architecture achieves the best performances so far with an accuracy of 78% and a SOV of 73.5%. A different approach [9, 10] uses bidirectional recurrent neural networks (BRNNs) for secondary structure prediction. BRNNs does not suffer of the limitations discussed above, so they do not need a filtering stage. This architecture achieves results equivalent to Jones' work. Lately, Hua and Sun [11] proposed the use of support vector machines (SVMs) for secondary structure prediction. The authors claim the superiority of this model, supported by an high value of SOV without the use of a filtering stage.

Given the work of Hua and Sun, we decided to explore the use of SVMs for the prediction of secondary structure. In section 2 we briefly explain the preparation of the data used during this work. In section 3 we test the use of SVMs for the prediction of secondary structure. We present here the results of an experiment run to replicate the claims made by Hua and Sun. Then, we apply the algorithm to a bigger and more representative dataset, posing attention on model selection. Finally, we compare SVMs and NNs on the same data. In section 4 we explore the use of bidirectional recurrent neural networks as a structure-to-structure

filtering classifier. In section 5 we present a novel method based on the Viterbi algorithm to enforce the prediction with constraints given in the form of a finite-state automaton (FSA). Finally, in section 6 we draw some conclusions about the results presented in this work, and we outline future directions of research inspired by these results.

## 2 Datasets

The first set of experiments is run to replicate the results of Hua and Sun [11]. In their work the authors used the publicly available dataset CB513 [12], composed by 513 chains with low similarity, so that test results are not biased. A 7-fold cross-validation is adopted to estimate the accuracy of the classifier. Evolutionary information is derived from multiple sequence alignments obtained from the HSSP database [13]. Secondary structure labels are assigned using the DSSP program [14].

The remainining of the experiments are performed using a significant fraction of the current representative set of non homologous chains from the Protein Data Bank (PDB Select [15]). We extracted the sequences from the April 2002 release, listing 1779 chains with a percentage of homology lower than 25%. From this set we retained only high quality proteins on which the DSSP program does not crash, determined only by X-ray diffraction, without any physical chain breaks and resolution threshold lower than 2.5 Å. The final dataset contains 969 chains, almost 184,000 amino acids, splitted in a training set of 490 chains, a validation set of 163 chains and a test set of 326 chains. Multiple alignments are generated using PSI-BLAST [16] applied to the Swiss-Prot+TrEMBL non-redundant database [17].

## 3 Support Vector Machines for Secondary Structure Prediction

The most successful predictors of secondary structure so far employ neural networks as classifiers. Lately, Hua and Sun [11] presented an SVM based architecture, claiming the superiority of this model as demonstrated by the high value of SOV reached. In this section we present our result about the use of SVMs for secondary structure prediction. We show the claimed value of SOV cannot be reached just implementing the classifier with SVMs. Moreover, we experiment SVMs on a bigger and more representative dataset to further exploit the potentiality of this model. Finally, we perform a direct comparison with neural networks, using the same data for both models.

### 3.1 SVM Classifier

Kernel machines and in particular support vector machines are motivated by Vapnik's principle of structural risk minimization in statistical learning theory [18]. In the simplest case, the SVM training algorithm starts from a vector-based representation of data points and searches a separating hyperplane that

has maximum distance from the dataset, a quantity known as the *margin*. More in general, when examples are not linearly separable vectors, the algorithm maps them into a high dimensional space, called *feature space*, where they are almost linearly separable. This is typically achieved via a kernel function that computes the dot product of the images of two examples in the feature space. The decision function associated with an SVM is based on the sign of the distance from the separating hyperplane:

$$f(\mathbf{x}) = \sum_{i=1}^{N} y_i \alpha_i K(\mathbf{x}, \mathbf{x}_i) \tag{1}$$

where $\mathbf{x}$ is the input vector, $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is the set of support vectors, $K(\cdot, \cdot)$ is the kernel function, and $y_i \in \{-1, 1\}$ is the class of the $i$-th support vector.

In their standard formulation SVMs output hard decisions, however, margins from equation 1 can be converted into conditional probabilities [19, 20, 21]. Platt [20] proposed to perform this mappping by mean of a logistic function, parameterized by an offset $B$ and a slope $A$. Parameters $A$ and $B$ are adjusted according to the maximum likelihood principle, assuming a Bernoulli model for the class variable. This can been extended [21] to the the multi-class case by assuming a multinomial model and replacing the logistic function by a softmax function [22]. More precisely, assuming $Q$ classes, we train $Q$ binary classifiers, according to the one-against-all output coding strategy. In this way, for each point $\mathbf{x}$, we obtain a vector $[f_1(\mathbf{x}), \cdots, f_Q(\mathbf{x})]$ of margins, that can be transformed into a vector of probabilities using the softmax function:

$$P(C = q|\mathbf{x}) = \frac{e^{A_q f_q(\mathbf{x}) + B_q}}{\sum_{r=1}^{Q} e^{A_r f_r(\mathbf{x}) + B_r}} \ , \ q = 1 \ldots Q \ . \tag{2}$$

The softmax parameters $A_q, B_q$ are determined as follows. First, we introduce a new dataset $\{(f_1(\mathbf{x}_i), \ldots, f_Q(\mathbf{x}_i), \mathbf{z}_i), i = 1, \ldots, m\}$ of examples whose input portion is a vector of $Q$ margins and output portion is a vector $\mathbf{z}$ of indicator variables encoding (in one-hot) one of $Q$ classes. As suggested by Platt for the two classes case, this dataset should be obtained either using a hold-out strategy, or a $k$-fold cross validation procedure. Second, we perform a search of the parameters $A_q$ and $B_q$ that maximize the log-likelihood function under a multinomial model:

$$\ell = \sum_{i}^{m} \sum_{q=1}^{Q} z_{q,i} \log P(C_i = q|\mathbf{x}) \tag{3}$$

where $z_{q,i} = 1$ if the $i$-th training example belongs to class $q$ and $z_{q,i} = 0$ otherwise.

### 3.2 Experiments on CB513

We now run a set of experiments to replicate the results of Hua and Sun [11] on the CB513 dataset. Our secondary structure predictor is constituted by three

**Table 1.** Results of the experiments on the CB513 dataset

| | $Q_3$ | $SOV$ |
|---|---|---|
| Our work | 73.2 | 68.5 |
| Hua and Sun | 73.5 | 76.2 |

one-against-all SVM classifiers with gaussian kernel combined using a softmax. We used the same parameters and the same inputs as in [11] in the attempt to replicate their best results. Our experimental results show a significant difference with respect to the value of SOV. This evidence supports our belief that the expected value of SOV reached by an SVM predictor should not be much different compared to a feed forward neural network approach, because both methods are *local*. There is no reason to expect that distinct models trained to predict a single position in the protein sequence and that achieve similar accuracy should behave completely different when their performance is measured on segments.

### 3.3 Experiments on PDB Select: SVM vs NN

The CB513 is a quite old dataset which is not representative of the current reality of the Protein Data Bank. Therefore, it is advisable to test the SVM classifier on a more significative dataset to better exploit its capabilities. Moreover, we want to perform an exstensive model search to find the optimal value of the $\gamma$ parameter of the gaussian kernel for each one-against-all classifier at various dimensions of the input window. For this purpouse, a small training set is used, because it would take too much time otherwise, while the error is estimated on a validation set. The value of $C$ is kept fixed to 1. The results of the model search (Table 2) show a saturation in the performances of the three classifiers for large windows. SVMs are capable of dealing with highly dimensional data, but they are unable to use the higher quantity of information contained in such richer inputs. It seems that augmenting the input window have the effect of increasing the quantity of noise more than the quantity of information.

Support vector machines require computationally expensive procedures for training. Therefore, we want to establish if they are somehow superior to neural networks in the task of secondary structure prediction. We use here the four layers feed forward neural network introduced by Riis and Krog [23]: an input

**Table 2.** Lowest errors achieved by each one-against-all classifier on the PDB select dataset optimizing the value of $\gamma$ at various dimensions $w$ of the input window

| Classifier | $w = 9$ | $w = 11$ | $w = 13$ | $w = 15$ | $w = 17$ | $w = 19$ |
|---|---|---|---|---|---|---|
| $H/\tilde{H}$ | 15.7 | 15.2 | 15.0 | **14.9** | 15.3 | 16.3 |
| $E/\tilde{E}$ | 16.4 | 16.0 | 15.7 | **15.5** | 15.6 | 15.7 |
| $C/\tilde{C}$ | 23.0 | 22.8 | **22.7** | 22.9 | 23.1 | 23.1 |

**Table 3.** Performances of the SVM architecture compared to the NN architecture on the PDB select dataset. Running time and size of the trained model are reported

| Method | $Q_3$ | $SOV$ | Time | Space |
|---|---|---|---|---|
| SVM | 76.5 | 68.9 | 3 days | 210 Mb |
| NN | 76.7 | 67.8 | 2 hours | 30 kb |

layer where a window of amino acids is fed, a code layer used for adaptively search an encoding of each amino acid, an hidden layer and an output layer where the prediction is taken. We applied both classifiers to the single split of the dataset described in section 2. The same input window is used for both the SVM and the NN architecture. The adaptive encoding layer of the NN has 3 neurons for each amino acid. The best NN model is searched using a validation set by varying the size of the hidden layer. The NN is trained with back-propagation and early stopping to avoid overfitting. The results of the experiments are shown in table 3. Both architectures reach a satisfying value of $Q_3$ but there is no clear advantage of the SVM over the NN model, while the first one have much higher time and space complexity. Given the very high number of examples and the possibility of using a validation-set, the prediction of secondary structure seems a task fitted for neural networks.

## 4 Filtering Predictions with Bidirectional Recurrent Neural Networks

Our experiments with SVM and NN architectures confirmed that a local classifier trained on single positions of the sequence cannot achieve a high value of SOV. The SOV is a very important measure to assess the quality of a classifier, since most of the uses of secondary structure predictions rely on the correct assignment of segments. Therefore, it is necessary to adopt an architecture which can correlate predictions on adjacent amino acids, to somehow "smooth" the final predicted sequence. In this work we explore the use of bidirectional recurrent neural networks as a filtering stage to refine the predictions of the local classifier.

BRNNs are recurrent neural networks where two set of states $F$ and $B$ are recursively copied forward and backward along the sequence (Figure 2). BRNNs can develop complex non-linear and non-causal dynamics that can be used to correct output-local prediction by trying to capture valid segments of secondary structure. Unfortunately, the problem of vanishing gradients [24] prevents from learning global dependencies, so it is impossible for the BRNN to model the whole conformation of the protein. The filtering BRNN has three inputs for each position of the sequence, corresponding to the probabilities calculated by the first stage classifier (Figure 2).

We used early stopping to control overfitting during training phase. We tested the BRNN on both the predictions of SVM and NN architectures. The experi-

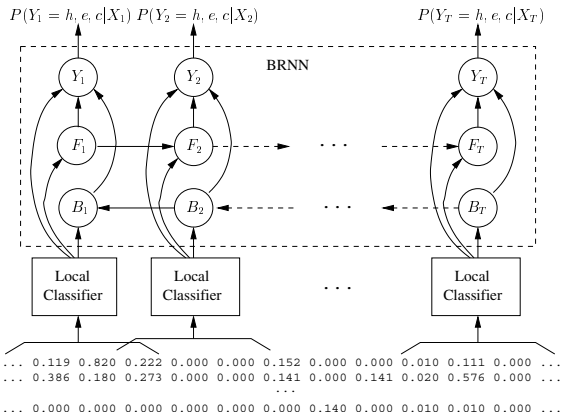



**Fig. 2.** Two stages architecture. Local classifier can be either SVM based or NN based. The bidirectional recurrent neural network is unfolded over the chain

**Table 4.** Performances of the various classifiers presented in this paper

| | $Q_3$ | $SOV$ |
|---|---|---|
| SVM | 76.5 | 68.9 |
| NN | 76.7 | 67.8 |
| SVM+BRNN | 77.9 | 74.1 |
| NN+BRNN | 77.8 | 74.2 |
| SVM+VD | 76.9 | 73.5 |
| NN+VD | 77.2 | 73.6 |
| SVM+BRNN+VD | 78.0 | 74.7 |
| NN+BRNN+VD | 78.0 | 75.2 |

ments (Table 4) clearly show the efficiency of the BRNN when used for filtering the predictions of a local classifier, with state-of-the-art accuracy and a very high value of SOV. The performances of this solution are equivalent to the architecture based on a BRNN with profiles as input [10], even if the filtering BRNN has a much simpler architecture and it is easier to train.

## 5 Enforcing Constraints Using the Viterbi Decoder

A close observation of the outputs of the two stages classifier shows the presence of *inconsistencies* in the predicted sequences. Alpha helices and beta strands in real proteins are identified by specific patterns of hydrogen bonds between amino acids. These way of labeling the secondary structure of a protein imposes some constraints on observable sequences:

- alpha helix segments must be at least 4 $\AA$ long,
- beta strands must be at least 2 $\AA$ long.

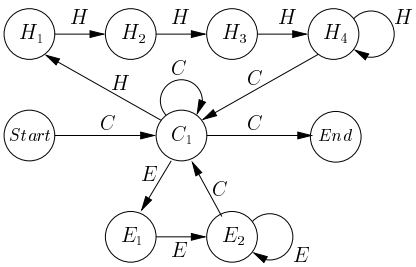



**Fig. 3.** Finite-state automaton representing every possible allowed sequence of secondary structure

some additional empirical facts enrich the list of constraints:

- a sequence must start and finish with a coil,
- between an alpha helix and a beta strand (and viceversa) there must be a coil.

We present here a simple but effective method to enforce constraints in the output of a classifier. All the facts known about physical chains can be expressed using a finite-state automaton (FSA, Figure 3), which represents every possible allowed sequence in our minimal secondary structure grammar. The outputs of the two stages classifier are the probabilities $P(H|X_t, t)$, $P(E|X_t, t)$ and $P(C|X_t, t)$ that the amino acid in position $t$ of the sequence is in one of the three secondary structure classes, given the input $X_t$ and the position $t$. We would like our constraints satisfying method to output the best possible sequence from the grammar defined by our FSA, using as a scoring function its overall probability:

$$P(Y|I) = \prod_{t=1}^{T} P(y_t|X_t, t)\,,\ Y = \{y_1 y_2 \dots y_T\}\ \ y_t \in \{H, E, C\}\ . \tag{4}$$

This request strictly resemble problem 2 of hidden Markov models [25]: we have the probabilities of observations, we have a state model of our data and we want the best sequence of states. A finite-state automaton can be thought of as a degenerated hidden Markov model, where each state emits a single symbol with probability 1, and all the transitions have the same probability. Therefore, we can employ the *Viterbi algorithm* to align our model to the sequence, using the probabilities of observations estimated by the classifier (Algorithm 1). The algorithm searches an optimal path on the nodes $(s, t)$ of the trellis, being $s$ the corresponding state of the FSA and $t$ the position in the sequence. Each node of the trellis has two attached variables: $score(s, t)$ is the score of the best sequence ending at this node, and $last(s, t)$ is the preceding state in the best sequence ending at this node. We define $symbol(s_i, s_j)$ as the symbol emitted during the transition from state $s_i$ to $s_j$, $parents(s)$ as the set states which have

a transition ending in state $s$, *start* is the set of starting states, and *end* is the best ending state. Log-probabilities are used because of numerical problems.

**Algorithm 1** The Viterbi decoder

*Init the trellis*: $\forall s, t: \ score(s,t) \leftarrow -\infty$

*Forward recursion*:
$score(start, 0) \leftarrow 0$
**for** $t = 1$ to $T$ **do**
  **for all** $s_i, s_j \in parents(s_i)$ **do**
    **if** $score(s_j, t-1) + \log P(symbol(s_i, s_j)|X_t, t) > score(s_i, t)$ **then**
      $score(s_i, t) \leftarrow score(s_j, t-1) + \log P(symbol(s_i, s_j)|X_t, t)$
      $last(s_i, t) \leftarrow s_j$
    **end if**
  **end for**
**end for**

*Backward recursion*:
$previous \leftarrow end$
**for** $t = T$ to $1$ **do**
  $this \leftarrow previous$
  $previous \leftarrow last(this, t)$
  $y_t \leftarrow symbol(previous, this)$
**end for**
$Y \leftarrow \{y_1 y_2 \ldots y_T\}$

The score of the ending state of the sequence is the log-probability of the best sequence $Y$. This algorithm can be applied to whatever FSA, and it represent a general way of imposing costraints on sequence of probabilities. In Table 4 we show the performances of the Viterbi decoder applied to the predictions of our classifiers. The Viterbi decoder can correct punctual errors, resulting in longer correct segments and higher values of SOV, even improving the predictions of the filtering stage.

## 6 Conclusions

In this paper we explored the use of SVMs for the prediction of secondary structure. We found that SVMs do not guarantee a high value of SOV, contrarily to a recent claim made by Hua and Sun [11]. Moreover, we found that SVMs are not superior to NN for secondary structure prediction when compared on the same data. Given the need of a filtering stage to refine the predictions of the local classifier and increase the value of SOV, we have explored the use of BRNNs for such task. We have demonstrated that a two stages architecture composed by a local classifier and a filtering BRNN can reach state-of-the-art performances. Finally,

we have introduced the Viterbi decoder to enforce constraints derived from prior knowledge into secondary structure predictions. The Viterbi decoder is capable of finding the best sequence of predictions from a predefined grammar given the probabilities estimated by a classifier. We have demonstrated the Viterbi decoder is able of increasing the value of SOV of our two stages architecture, and to output sequences which are consistent with the given constraints.

We have demonstrated that SVMs are not superior to other types of classifier for the problem of predicting secondary structure. The efficacy of SVMs is given by the possibility of working in high dimensionality spaces defined by kernels. The gaussian kernel does not constitute an improvement over the use of neural networks with sigmoid activation functions. The capabilities of SVMs would be really exploited if a more complex kernel using richer inputs is implemented. An example could be a kernel running directly on multiple alignments, without the need of calculating profiles which constitutes a loss in information. We have demonstrated the Viterbi decoder is able of correcting isolated errors, resulting in high values of SOV. However, the Viterbi decoder cannot correct completely misclassified segments of secondary structure. A solution to this problem could be the creation of a richer finite-state automaton, comprising constraints on sequence of secondary structure segments, even automatically discovered from observed structures.

## References

[1] Berman, H. M., Westbrook, J. , Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., Bourne, P. E.: The Protein Data Bank. Nucleic Acids Research **28**(2000):235–242 143

[2] Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., Thornton, J. M.: CATH - A Hierarchic Classification of Protein Domain Structures. Structure **5**(1997):1093–1108 144

[3] Murzin, A. G., Brenner, S. E., Hubbard, T., Chothia, C.: SCOP: a Structural Classification of Proteins Database for the Investigation of Sequences and Structures. Journal of Molecular Biology **247**(1995):563–540 144

[4] Qian, N., Sejnowski, T. J.: Predicting the Secondary Structure of Globular Proteins Using Neural Network Models. Journal of Molecular Biology **202**(1988):865–884 144

[5] Rost, B., Sander, C.: Prediction of Protein Secondary Structure at Better than 70% Accuracy, Journal of Molecular Biology **232**(1993):584–599 144

[6] Smith, T. F., Waterman, M. S.: Identification of Common Molecular Subsequences. Journal of Molecular Biology **147**(1981):195–197 144

[7] Zemla, A., Venclovas, C., Fidelis, K., Rost, B.: A Modified Definition of SOV, a Segment-Based Measure for Protein Secondary Structure Prediction Assessment. Proteins **34**(1999):220–223 144

[8] Jones, D. T.: Protein Secondary Structure Prediction Based on Position-Specific Scoring Matrices. Journal of Molecular Biology **292**(1999):195–202 144

[9] Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., Soda, G.: Exploiting the Past and the Future in Protein Secondary Structure Prediction. Bioinformatics **15**(1999):937–946 144

[10] Pollastri, G., Przybylski, D., Rost, B., Baldi, P.: Improving the Prediction of Protein Secondary Structure in Three and Eight Classes Using Recurrent Neural Networks and Profiles. Proteins **47**(2002):228–235 144, 149
[11] Hua, S., Sun, Z.: A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach. Journal of Molecular Biology **308**(2001):397–407 144, 145, 146, 147, 151
[12] Cuff, J. A., Barton, G. J.: Evaluation and Improvement of Multiple Sequence Methods for Protein Secondary Structure Prediction. Proteins **34**(1999)508–519 145
[13] Schneider, R., de Daruvar, A., Sander, C.: The HSSP Database of Protein Structure-Sequence Alignments. Nucleic Acids Research **25**(1997):226–230 145
[14] Kabsch, W., Sander, C.: Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. Biopolymers **22**(1983):2577–2637 145
[15] Hobohm, U., Sander, C.: Enlarged Representative Set of Protein Structures. Protein Science **3**(1994):522–524 145
[16] Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., Lipman, D. J.: Gapped BLAST and PSI-BLAST: a New Generation of Protein Database Search Programs. Nucleic Acids Research **25**(1997):3389–3402 145
[17] Bairoch, A., Apweiler, R., The Swiss-Prot Protein Sequence Data Bank and Its New Supplement TrEMBL. Nucleic Acids Research **24**(1996):21–25 145
[18] Vapnik, V.: Statistical Learning Theory. John Wiley, New York (1998) 145
[19] Kwok, J. T.: Moderating the Outputs of Support Vector Machine Classifiers IEEE Transactions on Neural Networks **10**(1999):1018–1031 146
[20] Platt, J.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D., eds: Advances in Large Margin Classifiers. MIT Press (1999) 146
[21] Passerini, A., Pontil, M., Frasconi, P.: From Margins to Probabilities in Multiclass Learning Problems. In F. van Harmelen, ed: Proc. 15th European Conf. on Artificial Intelligence. (2002) 146
[22] Bridle, J.: Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition Fogelman-Soulie, F., Hérault, J., eds: Neuro-computing: Algorithms, Architectures, and Applications. Springer-Verlag (1989) 146
[23] Riis, S. K., Krogh, A.: Improving Prediction of Protein Secondary Structure using Structured Neural Networks and Multiple Sequence Alignments. Journal of Computational Biology **3**(1996):163–183 147
[24] Bengio, Y., Simard, P., Frasconi, P.: Learning Long-Term Dependencies with Gradient Descent is Difficult. IEEE Transactions on Neural Networks **5**(1994):157–166 148
[25] Rabiner, L. R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE **77**(1989):257–286 150

# Adaptive Allocation of Data-Objects in the Web Using Neural Networks

Joaquín Pérez O.[1], Rodolfo A. Pazos R.[1], Hector J. Fraire H.[2], Laura Cruz R.[2], and Johnatan E. Pecero S.[2]

[1] National Center of Research and Technology Development
Cuernavaca, Mor. México
{jperez,pazos}@sd-cenidet.com.mx
[2] Ciudad Madero Technology Institute
Cd. Madero, Tam. México
hfraire@prodigy.net.mx
lcruzr@avantel.net
jpecero@yahoo.com

**Abstract.** In this paper we address the problem of allocation scheme design of large database-objects in the Web environment, which may suffer significant changes in usage and access patterns and scaling of data. In these circumstances, if the design is not adjusted to new changes, the system can undergo severe degradations in data access costs and response time. Since this problem is NP-complete, obtaining optimal solutions for large problem instances requires applying approximate methods. We present a mathematical model to generate a new object allocation scheme and propose a new method to solve it. The method uses a Hopfield neural network with the mean field annealing (MFA) variant. The experimental results and a comparative study with other two methods are presented. The new method has a similar capacity to solve large problem instances, regular level of solution quality and excellent execution time with respect to other methods.

## 1 Introduction

Currently many businesses are using Internet to connect their operations around the entire world. Many users are using mobile devices to access databases on the Web. User mobility produces frequent changes in access patterns. In this scenario the distributed database design and redesign is a critic problem for Web database administrators, and efficient tools and methodologies do not exists to facilitate their work [1, 13].

The logical design of a Web distributed database is an important and complicated task, which determines the data allocation in the system. If we consider that the nature of distributed systems is dynamic, with changes in the net topology, data access frequencies, cost and resources, the problem becomes even more complex.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 154–164, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Companies are incrementally investing in sophisticated software to implement their Web distributed databases. These systems are working now in a continually changing environment. They have to realize a large investment in technical staff to maintain adequately configured the software for responding to these changes. A major challenge for software designers of Web distributed database management systems is the automation of the allocation scheme design process. Distributed database management systems should incorporate an intelligent agent to automatically detect changes in the environment and to optimally reallocate the database objects. Adaptability will be an important characteristic for the software to work in future Web environments.

In [2] a mathematical model for this problem is proposed. The modeled problem is NP-complete [3] and therefore to obtain optimal (or suboptimal) solutions for large problem instances, heuristic methods are needed. In [12, 16, 17] three heuristic methods are proposed for solving this mathematical model. In this article a new solution method to solve it is proposed. The method uses a Hopfield neural network with the mean field annealing (MFA) variant. The experimental results and a comparative study with other two methods are presented. The new method has a similar capacity for solving large problem instances, regular level of solution quality, and excellent execution time with respect to other methods.

## 2 Mathematical Model of Database-Object Allocation

Traditionally it has been considered that the distributed database design consists of two sequential phases [1]. Contrary to this widespread belief, it has been shown that it is simpler to solve the problem in a single phase. The model presented in this paper follows the approach proposed in [2]. In order to describe the model and its properties, we introduce the following definition:

*Database-object*: Entity of the database that needs to be allocated, it can be an attribute, a record, a table or a file.

### 2.1 Description of the Allocation Problem

The model considers database-objects as independent units that must be allocated in the different sites of a network. The problem is to allocate objects, such that the total cost of data transmission for processing all the applications is minimized. A formal definition is the following:

Given a collection of objects $O = \{o_1, o_2, \ldots, o_n\}$, a computer communications network that consists of a collection of sites $S = \{s_1, s_2, \ldots, s_n\}$ where a series of queries $Q = \{q_1, q_2, \ldots, q_n\}$ are executed, and the access frequencies of each query from each site; the problem consists of obtaining a database-object allocation scheme that minimizes the transmission costs.

### 2.2 Description of the Allocation and Reallocation Problem

The model also allows generating new allocation schemes that adapt to changes in the usage and access patterns of read applications, thereby achieving the adap-

tation of the database to the dynamic characteristic of a distributed system and avoiding the system degradation. With this new characteristic of data migration, the problem is defined as follows:

Given a collection of objects $O = \{o_1, o_2, \ldots, o_n\}$, a computer communications network that consists of a collection of sites $S = \{s_1, s_2, \ldots, s_n\}$ where a series of queries $Q = \{q_1, q_2, \ldots, q_n\}$ are executed, an initial data-object allocation scheme, and the access frequencies of each query from each site. The problem consists of obtaining an allocation and reallocation optimized scheme that adapts to new database usage patterns and minimizes the transmission costs.

### 2.3 Objective Function

The integer (binary) programming model consists of an objective function and five intrinsic constraints. In this model the decision about storing an attribute $m$ in site $j$ is represented by a binary variable $x_{mj}$. Thus $x_{mj} = 1$ if $m$ is stored in $j$, and $x_{mj} = 0$ otherwise.

The objective function below (expression 1) models costs using four terms: 1) the required transmission cost to process all the queries; 2) the cost to access multiple remote objects required to execute the queries; 3) the cost for object storage in the sites, and 4) the transmission cost to migrate objects between nodes.

$$\min z = \sum_k \sum_i f_{ki} \sum_m \sum_j q_{km} l_{km} c_{ij} x_{mj} + \sum_i \sum_k \sum_j c1 f_{ki} y_{ki} + \sum_j c2 w_j + \sum_m \sum_i \sum_j a_{mi} c_{ij} d_m x_{mj} \tag{1}$$

where

$f_{ki}$ = emission frequency of query $k$ from site $i$, for a period of given time;
$q_{km}$ = usage parameter, $q_{km} = 1$ if query $k$ uses object $m$, otherwise $q_{km} = 0$;
$l_{km}$ = number of packets for transporting the items of object $m$ required for query $k$;
$c_{ij}$ = communication cost between sites $i$ and $j$;
$c1$ = cost for accessing several remote objects to satisfy a query;
$y_{kj}$ = indicates if query $k$ accesses one or more objects located at site $j$;
$c2$ = cost for allocating objects to a site;
$w_j$ = indicates if there exist objects at site $j$;
$a_{mi}$ = indicates if object $m$ was previously located at site $i$;
$d_m$ = number of packets for moving all the items of object $m$ to another site if necessary.

### 2.4 Problem Intrinsic Constraints

The model solutions are subject to five constraints: 1) each object must be stored in one site only, 2) each object must be stored in a site which executes at

least one query that uses it, 3 and 4) variables $w_j$ and $y_{kj}$ are forced to adopt values compatible with those of $x_{mj}$, and 5) site storage capacity must not be exceeded by the objects stored in each site. These constraints are formulated in the numbered expressions from 2 through 6.

$$\sum_j x_{mj} = 1 \quad \forall m \qquad \text{Each object must be stored in one site only.} \tag{2}$$

$$x_{mj} \leq \sum_k q_{km}\varphi_{ki} \quad \forall m, j \qquad \text{Each object } m \text{ must be stored in a site } j \text{ that executes at least one query involving the object; where } \varphi_{ki} = \begin{cases} 1 & \text{if } f_{ki} > 0 \\ 0 & \text{if } f_{ki} = 0 \end{cases} \tag{3}$$

$$t\, w_j - \sum_m x_{mj} \geq 0 \quad \forall j \qquad \text{This constraint forces the value of } w_j \text{ to 1 when any } x_{mj} \text{ equal 1, and induces } w_j \text{ to 0 otherwise, where } t = \text{number of objects.} \tag{4}$$

$$t\, y_{kj} - \sum_m q_{km} x_{mj} \geq 0 \quad \forall k, j \qquad \text{This constraint forces the value of } y_{kj} \text{ to 1 when any } q_{km}x_{mj} \text{ equal 1, and induces } y_{kj} \text{ to 0 otherwise.} \tag{5}$$

$$\sum_m x_{mj} p_m CA \leq cs_j \quad \forall j \tag{6}$$

The space occupied by all objects stored in site $j$ must not exceed the site capacity,where
$cs_i$ = capacity of site $j$;
$p_m$ = size in bytes of an item (tuple, attribute, instance, record) of object $m$;
$CA$ = cardinality of the object
= number of tuples, if the object is a relation or an attribute,
= number of records, if the object is a file.

## 3 Neural Network Method

### 3.1 Introduction

It is shown in [18] that the modeled problem is NP-complete. The demonstration consists of proving that the bin packing problem is reducible in polynomial time

to a subproblem of (1)–(6). For this reason, to solve large instance problems requires the use of approximate methods.

The model has been solved for small size problems using the Branch and Bound exact method [4]. For large size problems some approximate methods were used to solve it [3, 12, 15]. The approximate solution method that we propose in this work uses a Hopfield neural network with the mean field annealing (MFA) variant. In [6, 7, 8, 14] neural networks are used to solve complex optimization problems.

An approach for solving optimization problems using neural networks is based on statistical physics [5, 7]. Some related works are using the Hopfield model to solve optimization problems [5, 7, 8, 9]. In order to use a Hopfield neural network to solve an optimization problem, we need to incorporate the objective function and the constraints into the network model.

### 3.2 Energy Function

Formulating the energy function is not an easy task. The most frequent approach is to add the objective function and the constraints to the energy function using penalties. Each constraint is added as a penalty term. If $f(x)$ is the objective function and $C$ the constraints set, then the energy function takes the following form [6]:

$$E\left(\vec{x}\right) = m\, f\left(\vec{x}\right) + \sum_{c \in C} \lambda_{cp_c}\left(\vec{x}\right) \tag{7}$$

where $m = +1$ for minimization problems and $m = -1$ for maximization problems. The penalty term is zero if and only if the constraint is true.

### 3.3 Cost Function

The objective function of the model has four terms. The binary variable $x_{mj}$ indicates if object $m$ is stored in site $j$. This is an independent decision variable; however, decision variables $y_{kj}$ and $w_j$ are determined by the $x_{mj}$ values, through constraints (4) and (5).

The cost function $E(x)$ includes the first and fourth terms of the model objective function, because they involve the independent variable $x_{mj}$. The independent variable values are calculated using the neural net, and these are used to get the dependent variables values. Thus we can calculate the cost of the constructed solution. The problem is coded on the neural net, associating the binary variable values with the neurons values. This is carried out by substituting the model binary variable ($x_{mj}$) by the annealing mean field variable $v_{mj}$.

$$E\left(\vec{v_{mj}}\right) = \left(\sum_k \sum_i f_{ki} \sum_m \sum_j q_{km} l_{km} c_{ij} v_{mj} + \sum_m \sum_i \sum_j a_{mi} c_{ij} d_m v_{mj}\right) \tag{8}$$

### 3.4 Constraints

Constraints can be classified in two types: equal form and not equal form [14]. A method for modeling both types of constraints and constructing the penalty terms for the energy function is described in [6].

The constraints of the model are (2), (3), (4), (5) and (6), plus the condition that $x_{mj}$ must be a binary variable. Since constraints (4) and (5) depend on variables $w_j$ and $y_{kj}$, the energy function only uses constraints (2), (3) and (6). The penalty function is

$$P(v_{mj}) = \lambda_1 \sum_k \left( \sum_j v_{mj} - 1 \right)^2 + \lambda_2 \Omega \left( v_{mj} - \sum_k q_{km} \varphi_{ki} \right) + \lambda_3 \Omega \left( \sum_m x_{mj} p_m CA - cs_j \right) + \lambda_4 \sum_m \sum_j v_{mj}(1 - v_{mj}) \tag{9}$$

where the first three terms correspond to constrains (2), (3) and (6) respectively. The last term states that the independent variable must be binary. The $\Omega()$ function is defined as follows $\Omega(y) = y$ if $y \geq 0$ and $\Omega(y) = 0$ if $y < 0$. The penalty term is 0 if and only if the constraint is satisfied, otherwise the penalty is proportional to the violation degree.

### 3.5 Annealing Mean Field Technique

For applying the annealing mean field technique, it is necessary to replace the binary variable $s$ of the Hopfield model energy function by the average value of $s$ at a given temperature $T$:

$$\vec{v_i} = \left\langle \vec{s_i} \right\rangle_T \tag{10}$$

Using the Boltzman energy distribution, two alternative expressions for the $j$-th component of neuron $\vec{v_i}$ can be derived. In [7] the following expression is proposed:

$$v_{ij} = \frac{e^{u_{ij}/T}}{\sum_k e^{u_{ik}/T}} \tag{11}$$

In [5, 6, 7, 8] the following alternative expression is proposed:

$$v_{ij} = \frac{1}{2} \left[ 1 + \tanh \left( \frac{u_{ij}}{T} \right) \right] \tag{12}$$

where $u_{ij}$ is the local field and is given by

$$u_{ij} = -\frac{\partial E(v)}{\partial v_{ij}} \tag{13}$$

and the energy decrement is

$$\Delta E = \sum_j u_{mj} \left( {v_{mj}}^{new} - {v_{mj}}^{old} \right) \tag{14}$$

The basic steps for solving optimization problems with annealing mean field are [10, 11]:

- Code the problem in the Hopfield model energy function.
- Start with a high enough temperature value.
- While the temperature decreases, iteratively calculate the local field (Eq. 13) and update the neurons values (Eq. 11 or 12).
- Upon reaching neurons convergence ($\Delta E < \varepsilon$), the problem solution is decoded from the neurons values.
- The penalty parameters and the temperature reduction parameter can be tuned to improve the solution.

This approach, which uses the mean field neurons to code optimization problems on neural networks, is proposed in [7, 11].

Then, for our mathematical model the energy function is given by

$$\begin{aligned} E\left(\vec{v_{mj}}\right) = & \begin{pmatrix} \sum_k \sum_i f_{ki} \sum_m \sum_j q_{km} l_{km} c_{ij} v_{mj} + \\ \sum_m \sum_i \sum_j a_{mi} c_{ij} d_m v_{mj} \end{pmatrix} + \\ & \lambda_1 \sum_m \left( \sum_j v_{mj} - 1 \right)^2 + \lambda_2 \Omega \left( v_{mj} - \sum_k q_{km} \varphi_{ki} \right) + \\ & \lambda_3 \Omega \left( \sum_m x_{mj} p_m CA - cs_j \right) + \lambda_4 \sum_m \sum_j v_{mj} (1 - v_{mj}) \end{aligned} \tag{15}$$

To get the local field ($u_{ij}$) and the energy decrement $\Delta E$, (13) and (14) have to be used.

## 4 Method Implementation

### 4.1 Basic Algorithm

The basic algorithm [1, 11] that we are using for the implementation of the annealing mean field process is the following:

```
Initialize parameters. Set values for T=T0 and the decrement
parameter alpha; where
   0 < alpha < 1
Set Vmj with random values 0 <= Vmj <= 1
While T is in the freeze
range
```

```
  While E is decrementing
     Select a~random row m of Vmj;
     For j = 1, ..., JMAX
        Calculate the mean field Umj (eq. 4)
        Calculate the new spin Vmj.
        Calculate the energy change Delta E (eq. 5)
        Update the spin value Vmj^old = Vmj^new
     Repeat until Delta E < epsilon
Update T = alpha * T.
```

The initial value of the temperature is set at a temperature sufficiently high [12]. The value of the temperature decrement defines the speed of reduction of the temperature. The cooling process used was a geometric series, defined by $T = \alpha T$ with $0.85 < \alpha < 1$.

The penalty parameters were defined in terms of the penalization degree of the objective function, based on the number of violated constraints. The parameter values $\lambda_1$, $\lambda_2$, $\lambda_3$ were determined using an experimental procedure. The values used are $\lambda_1 = 500$, $\lambda_2 = 600$, $\lambda_3 = 500$.

### 4.2 Generation of Large Test Cases

Based on problems with known solution, a larger problem can be generated by adding components on the constraint matrix diagonal [12, 15]. Each component is a copy of the base problem. The solution of the problem generated this way is derived from the base problem solution.

For generating concatenate test cases the following procedure was used: let $P_1$ and $P_2$ denote two problem instances with known solution and $P_3$ denote a prob-

**Table 1.** Attributes, sites, queries and optimal value for each instance

| Instance | Attributes | Sites | Queries | Optimal Value |
|---|---|---|---|---|
| D1 | 10 | 4 | 8 | 5377.4 |
| D2 | 20 | 8 | 16 | 10754.8 |
| D3 | 40 | 16 | 32 | 21509.6 |
| D4 | 80 | 32 | 64 | 43019.2 |
| D5 | 90 | 36 | 72 | 48396.6 |
| D6 | 100 | 40 | 80 | 53774.0 |
| D7 | 120 | 48 | 96 | 64528.8 |
| D8 | 160 | 64 | 128 | 86038.4 |
| D9 | 170 | 68 | 136 | 91415.8 |
| D10 | 180 | 72 | 144 | 96793.2 |
| D11 | 200 | 80 | 160 | 107548.0 |
| D12 | 240 | 96 | 192 | 129057.6 |
| D13 | 320 | 128 | 256 | 172076.8 |

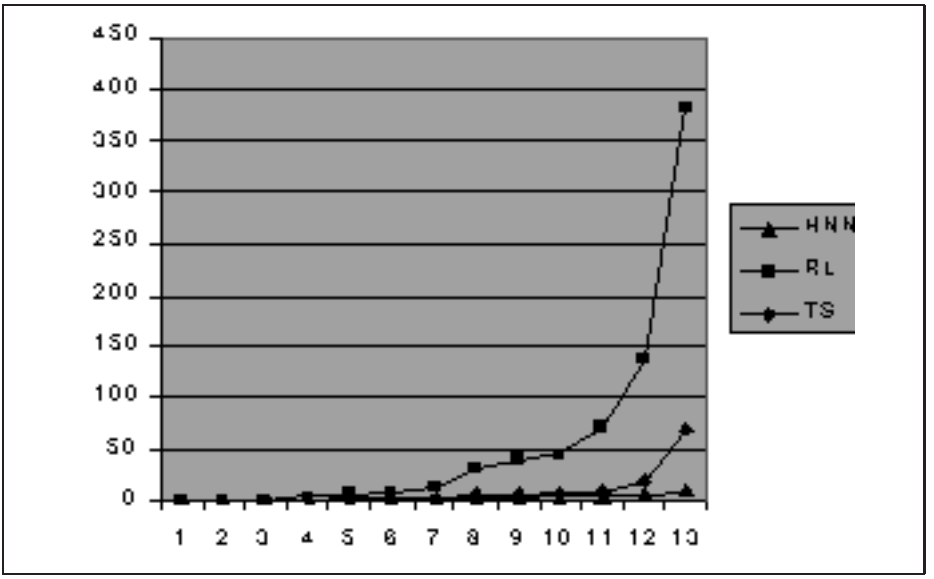



**Fig. 1.** Execution time graph

lem instance generated by concatenation. Then the solution of $P_3$ is the sum of the solutions of instances $P_1$ and $P_2$.

Table 1 shows the 13 instances used in the experiments, which were generated using the concatenation method. The solution of instance D1 was obtained using the software package LINDO; while the solutions of the remaining instances were obtained using the concatenation method.

### 4.3 Experiment Results

This section presents some of the experiment results of the Hopfield Neural Network (HNN) method. The goal is to evaluate the solution quality, execution time and the capacity of the proposed method for solving large problems.

Table 2 shows the comparative results of the application of the HNN method to the set of instances for which there exist experiment results obtained with two other approximate methods. The first column contains the instance identifier, while the second column shows the optimal solution for each instance. The remaining columns show for each method: the best value, percentage deviation with respect to the optimum, and average execution time. It is observed that all three methods show a similar average of percentage deviation: Reinforcement Learning (RL) [16] has 3.24, Neural Network has 4.02, and Tabu Search (TS) [17] has 4.27. The HNN method was much better than the other two methods regarding execution time (see Figure 1). For the largest instance D13 (320 attributes, 128 sites y 256 queries), the execution times for HNN, RL and TS were 10.356, 68.8 and 383.29 sec.

**Table 2.** Comparative results: Hopfield Neural Network (HNN), Reinforcement Learning (RL) and Tabu Search (TS) methods

| | | HNN | | RL | | TS | |
|---|---|---|---|---|---|---|---|
| I | Optimal z | Dev % | Avg. T (sec) | Dev % | Avg. T (sec) | Dev % | Avg. T (sec) |
| D1 | 5377.4 | 0.00 | 0.04 | 0.00 | 0.04 | 0.00 | 0.01 |
| D2 | 10754.8 | 0.00 | 0.09 | 0.00 | 0.16 | 0.00 | 0.03 |
| D3 | 21509.6 | 0.00 | 0.22 | 0.00 | 0.78 | 12.67 | 0.09 |
| D4 | 43019.2 | 2.41 | 0.67 | 2.39 | 4.68 | 2.63 | 0.72 |
| D5 | 48396.6 | 3.29 | 0.83 | 3.26 | 6.52 | 3.28 | 0.99 |
| D6 | 53774.0 | 4.00 | 1.03 | 3.96 | 8.61 | 4.15 | 1.32 |
| D7 | 64528.8 | 5.07 | 1.40 | 5.01 | 14.02 | 5.04 | 2.20 |
| D8 | 86038.4 | 6.17 | 2.41 | 5.61 | 32.16 | 5.61 | 5.25 |
| D9 | 91415.8 | 6.32 | 2.80 | 5.45 | 39.05 | 5.56 | 6.25 |
| D10 | 96793.2 | 6.46 | 3.09 | 5.31 | 46.19 | 5.31 | 7.47 |
| D11 | 107548.0 | 6.68 | 3.74 | 5.07 | 71.34 | 5.08 | 10.37 |
| D12 | 129057.6 | 6.32 | 5.56 | 3.96 | 138.95 | 3.96 | 20.52 |
| D13 | 172076.8 | 5.55 | **10.35** | 2.14 | **383.29** | 2.14 | **68.80** |
| | | | | | | | |
| Avg. Std. Dev % | | **4.02** | | **3.24** | | **4.25** | |

## 5 Conclusions

In this paper we present a mathematical model to generate new database object allocation schemes and a new method to solve it. Experimental data of the method performance were presented including a comparative study with two other methods.

The new method shows similar capacity for solving large problem instances, regular level of solution quality and excellent execution time with respect to the other two methods.

This work provides a new solution method for the mathematical model considered. For several years some methods have been tried on this model. Currently there exist methods for the model with simulated annealing, tabu search, reinforcement learning, genetic algorithms, and now neural nets. Model tests have been made with satisfactory and encouraging results. We consider that the principles used in the model are feasible for being used in other models for Web resource optimization.

Future work with this method includes: performance analysis with another kind of test cases and automatic setting of parameters.

## References


[1] Oszu, M., Valduries, P.: Principles of Distributed Database Systems. Prentice-Hall (1991) 553 154, 155, 160

[2] Pérez, J., Pazos, R., Frausto, J., Romero, D., Cruz, L.: Vertical Fragmentation and Allocation in Distributed Databases with Site Capacity Restrictions Using Threshold Accepting Algorithm. Lecture Notes in Artificial Intelligence 1793 (2000) 75–81 155

[3] Pérez, J., Pazos, R., Vélez, L., Rodríguez, G.: Automatic Generation of Control Parameters for the Threshold Accepting Algorithm. Lecture Notes in Artificial Intelligence 2313 (2002) 118–127 155, 158

[4] Pérez, J., Pazos, R., Romero, D., Santaolaya, R., Rodríguez, G., Sosa, V.: Adaptive and Scalable Allocation of Data-Objects in the Web. Lecture Notes in Computer Science 2667 (2003) 134–143 158

[5] Díaz, A., González, J., Laguna, M., Moscato, P.,Tseng, F., Glover, F., Ghaziri, F.: Optimización Heurística y Redes Neuronales, Editorial Paraninfo (1996) 235 158, 159

[6] Hilera, J., Martínez, V.: Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones. Editorial Alfaomega (1995) 390 158, 159

[7] Peterson, C., Söderberg, B.: Neural Optimization. 2th edn.The Handbook of Brain Research and Neural Networks, MIT Press (1995) 158, 159, 160

[8] Smith, K.: Neural Networks for Combinatorial Optimization: A review of More Than a Decade of Research. Journal of Computing, Vol. 11, No.1. (1999) 158, 159

[9] Smith K.: Neural Techniques for Combinatorial Optimization with Applications. IEEE Transactions on Neural Networks, Vol. 9, No. 6, (1998) 158

[10] Shaharuddin, S., Zomaya A.: Multiprocessor Scheduling Using Mean Field Annealing. http://ipdps.eece.unm.edu/1998/biosp3/bipp9.pdf 160

[11] Bultan T., Aykanat C.: A New Mapping Heuristic Based on Mean Field Annealing. Intel Supercomputer Systems Division and Research Council 160

[12] Vélez, L.: Esquema de Enfriamiento Adaptivo para el Algoritmo de Aceptación por Umbral Aplicado al Diseño de Bases de Datos Distribuidas. MS Thesis, Instituto Tecnológico de León, León (2000) 155, 158, 161

[13] Ceri, S., Pernici, B., Wiederhold, G.: Distributed Database Design Methodologies. Proc. IEEE, vol. 75, no. 5, (1987), 533–546 154

[14] M. Ohlsson, M., Peterson, C., Söderberg, B.: Neural Networks for Optimization Problems with Inequality Constraints- the Knapsack Problem. Department of Theorical Physics, University of Lund, Sweden 158, 159

[15] Colunga, A., Muñoz, L., Ramírez, M.: Agente Inteligente de Aprendizaje Reforzado Aplicado al Diseño de Bases de Datos Distribuidas. BS Thesis, Instituto Tecnológico de Cd. Madero, (2001) 158, 161

[16] González, J., Fraire, H., Moreno, A., Muñoz, L. Ramírez, M.: Agente Inteligente de Aprendizaje Reforzado Aplicado al Diseño de Base de Datos Distribuidas. Procs. IEEE, CIECE (2001) 155, 162

[17] Laurence, C.: Evaluación del Modelo DFAR Usando Tabu Search. Technical report, Instituto Tecnológico de Cd. Madero (2001) 155, 162

[18] Pérez, J., Pazos, R. A., Romero, D., Cruz, L.: Análisis de Complejidad del Problema de la Fragmentación Vertical y Reubicación Dinámica en Bases de Datos Distribuidas. Proceeding of 7mo. Congreso Internacional de Investigación en Ciencias Computacionales (2000) 157

# Improving the SLA Algorithm Using Association Rules

Evelina Lamma[1], Fabrizio Riguzzi[1], Andrea Stambazzi[1], and Sergio Storari[1,2]

[1]University of Ferrara, Department of Engineering
via Saragat 1, 44100 Ferrara, Italy
[2]University of Bologna, DIES
viale Risorgimento 2, 40136 Bologna, Italy
{elamma,friguzzi,astambazzi,sstorari}@ing.unife.it

**Abstract.** A bayesian network is an appropriate tool for working with uncertainty and probability, that are typical of real-life applications. In literature we find different approaches for bayesian network learning. Some of them are based on search and score methodology and the others follow an information theory based approach. One of the most known algorithm for learning bayesian network is the SLA algorithm. This algorithm constructs a bayesian network by analyzing conditional independence relationships among nodes. The SLA algorithm has three phases: drafting, thickening and thinning. In this work, we propose an alternative method for performing the drafting phase. This new methodology uses data mining techniques, and in particular the computation of a number of parameters usually defined in relation to association rules, in order to learn an initial structure of a bayesian network. In this paper, we present the BNL-rules algorithm (Bayesian Network Learner with association rules) that exploits a number of association rules parameters to infer the structure of a bayesian network. We will also present the comparisons between SLA and BNL-rules algorithms on learning four bayesian networks.

## 1 Introduction

A bayesian network is an appropriate tool for working with uncertainty and probability, that are typical of real-life applications. A bayesian network is a directed, acyclic graph (DAG) whose nodes represent random variables. In bayesian networks each node, V, is conditionally independent of any subset of the nodes that are not its descendants given its parents. Bayesian networks are sometimes called causal networks because the arcs connecting the nodes can be thought of as representing direct causal relationships. Building bayesian networks on the basis of the intuitive (human) notion of causality usually results in networks that respect the conditional independence assumptions. According to [12] "… to construct a bayesian network for a given set of variables, we draw arcs from cause variables to immediate effects. In almost all cases, doing so results in a Bayesian network whose conditional-independence implications are accurate".

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 165-175, 2003.
© Springer-Verlag Berlin Heidelberg 2003

By means of bayesian networks, we can use information about the values of some variables to obtain probabilities for the values of others. A probabilistic inference takes place, once the probabilities functions of each node conditioned to just its parents are given. These are usually represented in a tabled form, named conditional probability tables (CPTs).

Learning techniques have been extensively applied (see, for instance [12]) in Bayesian networks. Given a training set of examples, learning such a network is the problem of finding the structure of the direct acyclic graph and the CPTs associated with each node in the DAG that best matches (according to some scoring metric) this dataset. Various scoring metrics have been proposed (e.g., description length or posterior probability [2,17,20,10,12,13,14,25]). Learning algorithms perform a search among possible network structures, however the search space is so vast, that any kind of exhaustive search can not be considered, and often a greedy approach is followed.

One of the most known algorithm for learning bayesian network is the SLA algorithm presented in [8]. This algorithm constructs a bayesian network by analyzing conditional independence relationships among nodes. It computes the mutual information of each pair of nodes as a measure of dependency, and creates the network using this information. The SLA algorithm has three phases: drafting, thickening and thinning. The first phase of this algorithm is essentially Chow and Liu's tree construction algorithm [9]; the second and the third phase are designed to extend tree construction to general bayesian network construction. The draft is a singly connected undirected graph (an undirected graph without loops). In a special case when the bayesian network is a tree or polytree, this phase can construct the network correctly and the second and third phase will not change anything.

In this work, we propose another way for performing the drafting phase. This new methodology uses data mining techniques, and in particular the computation of a number of association rules parameters given a database of examples, in order to learn the structure of a bayesian network. Association rules describe correlation of events, and can be viewed as probabilistic rules. Two events are “correlated” if they are frequently observed together. Each association rule, is characterized by several parameters which can be used in structure learning. In this paper, we present the BNL-rules algorithm (Bayesian Network Learner with association rules) that exploits these parameters to infer the structure of a bayesian network.

Section 2 discusses the conditional independency test used by the SLA algorithm. Section 3 describes the SLA algorithm itself. In Section 4, we present association rules parameters. In Section 5 we introduce the algorithm BNL-rules. In Section 6, we will present the comparisons between SLA and BNL-rules algorithms considering four of the most well-known bayesian networks. Finally, in Section 7, we conclude, and present future work.

## 2 An Information Theory Based Approach

In literature we find different approaches for bayesian network learning. Some of them are based on the search and score methodology [2,17,20,10,12,14,25], and the others follow an information theory based approach [8,21]. The SLA algorithm constructs bayesian networks by analyzing conditional independence relationships

among nodes, following the second approach. Before introducing the SLA algorithm, we first recall the concepts of *d-separation* [17] and mutual information [8], which play an important role in this algorithm.

For any three disjoint node sets $X$, $Y$, and $Z$ in a bayesian network, $X$ is said to be *d-separated* from $Y$ by $Z$ if there is no *active adjacency* path between $X$ and $Y$ given $Z$. An *adjacency* path is a path between two nodes without considering the directionality of the arcs. An *adjacency* path between $X$ and $Y$ is *active* given $Z$ if:

- every *collider* in the path is in $Z$ or has a descendant in $Z$;
- every other node in the path is outside $Z$.

A *collider* [23] of a path is a node where two arcs in the path meet at their endpoints. In a bayesian network, if there is an arc from $a$ to $b$, we say that $a$ is a parent of $b$ and $b$ is a child of $a$. We also say that $a$ is in the neighborhood of $b$ and $b$ is in the neighborhood of $a$. If there is a path from node $a$ to node $b$, then $a$ is an ancestor of $b$ and $b$ is a descendant of $a$.

To understand *d-Separation*, we can use an analogy, which is similar to the one suggested in [24]. We view a bayesian network as a network system of information channels, where each node is a valve that is either active or inactive and the valves are connected by noisy information channels. The information flow can pass an active valve but not an inactive one. When all the valves (nodes) on one *adjacency* path between two nodes are active, we say this path is **open**. If any one valve in the path is inactive, we say the path is **closed**. When all paths between two nodes are **closed** given the statuses of a set of valves (nodes), we say the two nodes are *d-separated* by the set of nodes. The statuses of valves can be changed through the instantiation of a set of nodes.

The amount of information flow between two nodes can be measured by using mutual information when no nodes are instantiated, or conditional mutual information when some other nodes are instantiated. In information theory, the mutual information of two nodes is defined as:

$$I\left(X_i, X_j\right) = \sum_{x_i, x_j} P\left(x_i, x_j\right) \log \frac{P\left(x_i, x_j\right)}{P\left(x_i\right) P\left(x_j\right)} \tag{1}$$

where $X_i$ and $X_j$ are two nodes and $x_i$ and $x_j$ are possible values for $X_i$ and $X_j$. The conditional mutual information is defined as

$$I\left(X_i, X_j \mid C\right) = \sum_{x_i, x_j, c} P\left(x_i, x_j, c\right) \log \frac{P\left(x_i, x_j \mid c\right)}{P\left(x_i \mid c\right) P\left(x_j \mid c\right)} \tag{2}$$

where $C$ is a set of nodes and $c$ is a possible assignment of all the nodes in $C$. In the SLA algorithm, the conditional mutual information measures the average information flow between two nodes when the statuses of some valves are changed by the condition-set $C$. When equation (2) is smaller than a certain threshold value $\varepsilon$, we say that $X_i$, $X_j$ are *d-separated* by the condition-set $C$, and that they are conditionally independent. Thus, the mutual information is a measure of Conditional Independence (CI). The use of mutual information in probabilistic model construction can be traced back to Chow and Liu's tree construction algorithm [9]. In 1987, Rebane and Pearl

extended Chow and Liu's algorithm to causal polytree construction [19]. The SLA algorithm extends those algorithms further to Bayesian network construction. This algorithm also makes the following two assumptions: the database attributes have discrete values and there are no missing values in all the records; the volume of data is large enough for reliable CI tests.

## 3 The SLA Algorithm

The SLA algorithm has three phases: drafting, thickening and thinning. The drafting phase of this algorithm is essentially Chow and Liu's tree construction algorithm; the thickening and thinning phase are designed to extend tree construction to general bayesian network construction. In the drafting phase, this algorithm computes the mutual information of each pair of nodes as a measure of independency, and creates a draft based on this information. The draft is a singly connected undirected graph (a graph without loops). In a special case, when the bayesian network is a tree or polytree, this phase can construct the network correctly and the second and third phase will not change anything. In the thickening phase, the algorithm adds edges when the pairs of nodes cannot be *d-separated*. The result of this phase has the structure of an *independence map* (*I-map*) [19] of the underlying dependency model in the case that the underlying model is *normal DAG-Faithful* [8]. In the thinning phase, each edge of the *I-map* is examined using CI tests and is removed if the two nodes of the edge can be *d-separated*. The result of this phase has the structure of a *perfect map* [19] when the underlying model is *normal DAG-Faithful*. At the end of the third phase, the SLA algorithm also carries out a procedure to orient the edges of the graph. The basics of the three phases are reported in Figure 1, Figure 2 and Figure 3.

The SLA algorithm uses the procedures: **try_to_separate_A**, **try_to_separate_B** and **orient_edges** (see Figure 2 and Figure 3). Given a graph and two nodes *a* and *b*, the procedure **try_to_separate_A** tries to identify if these two nodes are *d-separated.* From the definition of bayesian network [17] we know that if two nodes *a, b* in the network are not connected, they can be *d-separated* by the parent nodes of *b* which are in the paths between those two nodes. (We assume that node *a* appears earlier in the node ordering than *b*.) Those parent nodes form a set *P*. If node ordering is known, we can get *P* immediately and only one CI test is required to check if two nodes are *d-separated*. Since this information is usually not given, we have to use a group of CI tests to find such *P*. By assuming that removing a parent node of *b* will not increase the mutual information between *a* and *b*, the above procedure tries to find set *P* by identifying and removing the child-nodes and irrelevant nodes from the set of neighbors of *a* and the set of neighbors of *b* one at a time using a group of computations and comparisons of conditional mutual information. However, this assumption may not be true when the underlying structure satisfies the following conditions:

1. There exists at least one path from *a* to *b* through a child-node of *b* and this child-node is a *collider* on the path.
2. In such paths, there are one or more *colliders* besides the child-node and all these *colliders* are the parents or ancestors of *b*.

The drafting phase is composed by five steps:

D.1. Initiate a graph $G(V, E)$ where $V$={all the nodes of a data set}, $E$={ }. Initiate an empty list $L$.

D.2. For each pair of nodes $(v_i, v_j)$ where $v_i, v_j \in V$, compute mutual information $I(v_i, v_j)$ using equation (1). For all the pairs of nodes that have mutual information greater than a certain small value $\varepsilon$, sort them by their mutual information and put these pairs of nodes into list $L$ in decreasing order. Create a pointer $p$ that points to the first pair of nodes in $L$.

D.3. Get the first two pairs of nodes of list $L$ and remove them from it. Add the corresponding edges to $E$. Move the pointer $p$ to the next pair of nodes.

D.4. Get the pair of nodes from $L$ at the position of the pointer $p$. If there is no *adjacency* path between the two nodes, add the corresponding edge to $E$ and remove this pair of nodes from $L$.

D.5. Move the pointer $p$ to the next pair of nodes and go back to step D.4 unless $p$ is pointing to the end of $L$ or $G$ contains $n$-1 edges. ($n$ is the number of nodes in G.)

**Fig. 1.** Drafting the network

The thickening phase is composed by three steps:

Tk.1. Move the pointer $p$ to the first pair of node in $L$. Get the pair of nodes from $L$ at the position of the pointer $p$.

Tk.2. Call procedure **try_to_separate_A(current graph, node1, node2)** to see if this pair of nodes can be separated in current graph. If so, go to next step; otherwise, connect the pair of nodes by adding a corresponding edge to $E$. (Procedure **try_to_separate_A** will be presented later in this subsection.)

Tk.3. Move the pointer $p$ to the next pair of nodes and go back to step Tk.2 unless $p$ is pointing to the end of $L$.

**Fig. 2.** Thickening the network

The thinning phase is composed by three steps:

Th.1. For each edge in $E$, if there are other paths besides this edge between the two nodes, remove this edge from $E$ temporarily and call procedure **try_to_separate_A (current graph, node1, node2)**. If the two nodes are dependent, add this edge back to $E$; otherwise remove the edge permanently.

Th.2. For each edge in $E$, if there are other paths besides this edge between the two nodes, remove this edge from $E$ temporarily and call procedure **try_to_separate_B (current graph, node1, node2)**. If the two nodes are dependent, add this edge back to $E$; otherwise remove the edge permanently. (Procedure **try_to_separate_B** will be presented later in this subsection.)

Th.3. Call procedure **orient_edges (current graph)**. (This procedure will be presented later in this subsection.)

**Fig. 3.** Thinning the network

In such structures, procedure **try_to_separate_A** may identify a parent-node of $b$ as a child-node of $b$ and remove it erroneously. As a result, the procedure fails to separate two *d-separated* nodes. To deal with these structures, a correct procedure **try_to_separate_B** is introduced. Theoretically, is possible to use procedure **try_to_separate_B** to replace procedure **try_to_separate_A,** since they do the same thing and both of them have complexity $O(N^4)$ on CI test. But in practice, procedure **try_to_separate_A** usually uses fewer CI tests and requires smaller condition-sets. Therefore SLA tries to avoid using procedure **try_to_separate_B** whenever it is possible.

Among the nodes in Bayesian networks, only *colliders* can let information pass through them when they are instantiated. The **orient_edges** procedure uses this feature to identify *colliders*. All other edge orientations are virtually based on these identified *colliders*. The *collider* based edge orientation methods have also been studied in [19,23].

## 4 Association Rules

Association rules describe correlation of events and can be regarded as probabilistic rules. Events are "correlated" if they are frequently observed together. For example, in the case of sale transactions, an event is the sale of a particular product and association rules express which items are usually bought together.

Consider a database $D$ consisting of a single table. An association rule [1] is a rule of the form

$$A_1=v_{A1},\ A_2=v_{A2},\ \ldots,\ A_j=v_{Aj} \Rightarrow B_1=v_{B1},\ B_2=v_{B2},\ \ldots,\ B_k=v_{Bk}$$

where $A_1, A_2, \ldots, A_j, B_1, B_2, \ldots, B_k$ are attribute of $D$ and $v_{A1}, v_{A2}, \ldots, v_{Aj}, v_{B1}, v_{B2}, \ldots, v_{Bk}$ are values such that $v_{Ai}$ ($v_{Bh}$) belongs to the domain of the attribute $A_i$ ($B_h$).

More formally, an association rule can be defined as follows.
An *item* is a literal of the form $A_i=v_{Ai}$ where $A_i$ is an attribute of $D$ and $v_{Ai}$ belongs to the domain of $A_i$. Let $I$ be the set of all the possible items. A *transaction* $T$ is a record of $D$.

An *itemset* $X$ is a set of items, i.e. it is a set $X$ such that $X \subseteq I$. We say that a transaction $T$ *contains* an itemset $X$ if $X \subseteq T$ or, alternatively, if $T$ satisfies all the literals in $X$.

The *support* of an itemset $X$ (indicated by *support(X)*) is the fraction of transactions in $D$ that contain $X$. The support of the opposite of an itemset (indicated by *support(!X)*) is the fraction of transactions in $D$ that do not contain $X$. Thus, *support(!X)*= 1-*support(X)*.

An *association rule* is an implication of the form $X \Rightarrow Y$, where $X$ and $Y$ are itemsets and $X \cap Y \neq \emptyset$. The *support* of $X \Rightarrow Y$ (indicated by *support(X⇒Y)*) is *support(X ∪ Y)*. The *confidence* of $X \Rightarrow Y$ (indicated by *confidence(X⇒Y)*) is the fraction of transactions in $D$ containing $X$ that also contain $Y$. Thus, *confidence(X⇒Y) = support(X ∪ Y) / support(X)*.

Given an association rule $X \Rightarrow Y$, we are interested in the following parameters:

- The lift [4] (called interest in [6]) of X ⇒ Y (indicated by lift(X⇒Y)) is given by lift(X⇒Y) = confidence(X⇒Y) / support(Y). Thus, lift(X⇒Y) = support(X∪Y) / (support(X) × support(Y)).
- The leverage [26] of X ⇒ Y (indicated by leverage(X⇒Y)) is given by leverage(X⇒Y) = support(X ∪ Y) - support(X) × support(Y).
- The conviction [6] of X ⇒ Y (indicated by conviction(X⇒Y)) is given by conviction(X⇒Y) = support(X) × support(!Y) / support(X ∪ !Y). Observing that support(X∪!Y) = support(X) - support(X∪Y), we have that conviction(X⇒Y) =(1-support(Y)) / (1-confidence(X⇒Y)).

Moreover, other parameters can be defined that regard not a single association rule but a set of specific association rules, namely, the set of association rules that relate two variables. Supposing that we have binary variables $X$ and $Y$, such a set of association rules would be $\{X=0 \Rightarrow Y=0,\ X=1 \Rightarrow Y=0,\ X=0 \Rightarrow Y=1,\ X=1 \Rightarrow Y=1\}$ plus the rules with head and body exchanged.

These parameters are Person's $X^2$ and the Cramer index [11]. Person's $X^2$ has been proposed as an interesting parameter for association rules in [7] and [22]. In order to

define Person's $X^2$ and the Cramer index, let us consider two variables $X$ and $Y$ where $X$ can assume $I$ different values $x_1,...x_I$ and $Y$ can assume $J$ different values $y_1,...y_J$. Moreover, let us define the following parameters: $n=|D|$, $n_{ij}=support(\{X=x_i,Y=y_j\})n$, $n_{i\bullet}=support(\{X=x_i\})n$, $n_{\bullet j}=support(\{Y=y_j\})n$ and $n^*_{ij}=n_{i\bullet}n_{\bullet j}/n$. $X^2$ is then given by

$$X^2 = \sum_{i=1}^{I}\sum_{j=1}^{J}\frac{(n_{ij}-n^*_{ij})^2}{n^*_{ij}} \tag{3}$$

The Cramer Index $V$ is instead given by

$$V = \sqrt{\frac{X^2}{n\min\{(I-1),(J-1)\}}} \tag{4}$$

$X^2$ is a statistics proposed by Karl Pearson for verifying the hypothesis of stochastic independence between two variables $X$ and $Y$. $X^2$ is 0 if $X$ and $Y$ are independent, while the higher it is the less probable it is that the two variables are independent. The Cramer Index scales the value of $X^2$ to the maximum that can assume in the table examined, therefore $0 \le V \le 1$. If $V=1$ we have the maximum dependence between $X$ and $Y$.

## 5 BNL-Rules Algorithm

In this section, we propose a novel method for performing the drafting phase of the SLA algorithm by using association rules parameters. The algorithm we propose is called BNL-rules (Bayesian Network Learner with association rules).

During our experiments we tried to use both the most specific association rules and the most general ones. The most specific rules demonstrated to be unuseful for our task because they were not able describe the single dependency/independency relation between two variables, so we decided to use the most general ones, i.e., those with only one antecedent and only one consequent (named one-to-one rules). Each rule is characterized by the typical association rule parameters described in Section 3 (lift, conviction and leverage). Moreover, we associate to each such rule the values of the $X^2$ and Cramer Index parameters relative to the couple of variables involved in the rule.

The BNL-rules algorithm considers all the possible one-to-one rules and computes for all of them the values of the parameters. Then it sorts the rules in descending order of one of the parameters and builds a draft of the network: if the network has N nodes, the tool creates a network connecting the nodes indicated in the rules, without introducing loops, until it reaches N-1 arcs. In this way we obtain an undirected acyclic graph. At the moment we haven't identified a method for learning arc orientations by using association rule parameters.

A formal description of the BNL-rules algorithm is presented below.

**BNL-Rules Algorithm**
Given a set of examples described by N variables, the bayesian network drafting performed by BNL-rules is realized in in six steps:

D.1. Create a graph *G(V, E)* where *V*={all the nodes of a data set}, *E*={ }. Create an empty list *L*.
D.2. For each one-to-one rule compute the parameters. Each rule and associated parameters become an element of list *L*.
D.3. Choose one of the parameters (lift, leverage, convinction, $X^2$ or Cramer Index), and sort the elements of *L* in decreasing order with respect to this parameter.
D.4. Get the first element of list *L* and remove it from the list. Add the corresponding edges to *E*. Move the pointer *p* to the next element of *L*.
D.5. Get the element from *L* at the position of the pointer *p*. If there is no *adjacency* path between the two nodes, add the corresponding edge to *E* and remove this element from *L*.
D.6. Move the pointer p to the next element and go back to step D.5 unless p is pointing to the end of L or G contains N-1 edges.

This algorithm is similar to the one proposed by SLA for the drafting phase, but instead of computing the mutual information, the algorithm computes the parameters of all the possible one-to-one rules.

## 6 Comparison between SLA and BNL-Rules Algorithms

We experimented the algorithms with four different bayesian networks:

- The "Visit to Asia" network: A belief network for a fictitious medical example about whether a patient has tuberculosis, lung cancer or bronchitis, related to their X-ray, dyspnea, visit-to-Asia and smoking status. It has 8 nodes and 8 arcs. It is described in [16].
- The "Car_diagnosis" network: A belief network for diagnosing why a car won't start, based on spark plugs, headlights, main fuse, etc. It has 18 nodes and 20 arcs. It is described in [18];
- The "ALARM" network: ALARM stands for "A Logical Alarm Reduction Mechanism". This is a medical diagnostic system for patient monitoring. It is a nontrivial belief network with 8 diagnoses, 16 findings and 13 intermediate variables (36 nodes and 46 arcs). It is described in [3];
- The "Boelarge92" network: A subjective belief network for a particular scenario of neighborhood events, that shows how even distant concepts have some connection. It has 24 nodes and 35 arcs. It is described in [5];

A database of examples was generated from each network using NETICA [18]. Then SLA and BNL-rules are applied to learn back the network. The learned network is compared with the original and the numbers of missing and extra arcs are counted. Tables 1, 2, 3 and 4 show these comparison results for each network, for each algorithm and for three different dataset dimensions (5.000, 20.000 and 100.000 examples).

**Table 1.** Results for the "Visit to Asia" network

| Data Set | SLA Drafting | | BNL-rules Leverage | | BNL-rules Convinction | | BNL-rules Lift | | BNL-rules Pearson's $X^2$ | | BNL-rules Cramèr Index | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra |
| 5000 | 1 | 0 | 3 | 2 | 3 | 2 | 3 | 2 | 1 | 0 | 1 | 0 |
| 20000 | 1 | 0 | 3 | 2 | 3 | 2 | 3 | 2 | 1 | 0 | 1 | 0 |
| 100000 | 1 | 0 | 3 | 2 | 4 | 3 | 3 | 2 | 1 | 0 | 1 | 0 |

**Table 2.** Results for the "Car diagnosis" network

| Data Set | SLA Drafting | | BNL-rules Leverage | | BNL-rules convinction | | BNL-rules Lift | | BNL-rules Pearson's $X^2$ | | BNL-rules Cramèr Index | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra |
| 5000 | 4 | 1 | 5 | 2 | 9 | 6 | 10 | 7 | 5 | 2 | 5 | 2 |
| 20000 | 4 | 1 | 3 | 0 | 7 | 4 | 8 | 5 | 3 | 0 | 4 | 1 |
| 100000 | 4 | 1 | 3 | 0 | 7 | 4 | 8 | 5 | 3 | 0 | 4 | 1 |

**Table 3.** Results for the "Alarm" network

| Data Set | SLA Drafting | | BNL-rules Leverage | | BNL-rules Convinction | | BNL-rules Lift | | BNL-rules Pearson's $X^2$ | | BNL-rules Cramèr Index | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra |
| 5000 | 12 | 2 | 13 | 3 | 13 | 3 | 13 | 3 | 11 | 1 | 13 | 3 |
| 20000 | 12 | 2 | 11 | 1 | 15 | 5 | 12 | 2 | 11 | 1 | 13 | 3 |
| 100000 | 12 | 2 | 11 | 1 | 13 | 3 | 10 | 0 | 11 | 1 | 13 | 3 |

**Table 4.** Results for the "Boerlage" network

| Data Set | SLA Drafting | | BNL-rules Leverage | | BNL-rules Convinction | | BNL-rules Lift | | BNL-rules Pearson's $X^2$ | | BNL-rules Cramèr Index | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra | Miss | Extra |
| 5000 | 14 | 0 | 14 | 0 | 15 | 1 | 16 | 2 | 14 | 0 | 15 | 1 |
| 20000 | 14 | 0 | 14 | 0 | 14 | 0 | 16 | 2 | 14 | 0 | 14 | 0 |
| 100000 | 14 | 0 | 14 | 0 | 14 | 0 | 16 | 2 | 14 | 0 | 14 | 0 |

In order to test whether the tool used for generating the dataset has an influence over the results, we repeated the experiments with datasets generating the database with HUGIN [15]. The results obtained are identical except for the "Alarm" network drafted by BNL-rules employing leverage and a dataset of 100,000 examples that, in the case of HUGIN, has 12 missing arcs and 2 extra arc. Therefore the results are sufficiently independent from the database generation tool.

From the tables we can observe that BNL-rules with Pearson's $X^2$ always obtains the best results when used with 20,000 and 100,000 examples, outperforming SLA in two cases and performing equally well in the other two cases. When used with 5,000 examples, BNL-rules with Pearson's $X^2$ does worse than SLA only for the "Car diagnosis" network.

BNL-rules with leverage obtains results that are as good as those obtained with $X^2$ for all networks and for 20.000 and 100.000 examples apart from "Visit to Asia".

However, note that "Visit to Asia" is very small and thus the results on it are less significative than those on the others.

Note also that while the performances of SLA do not improve with the number of examples, those of BNL-rules with $X^2$ improve in one case and those of BNL-rules with leverage improve in two cases when going from 5.000 to 20.000 examples.

## 7 Conclusions

In this work we describe a method for improving SLA [8], one of the most known algorithm for learning Bayesian network, by exploiting a number of association rules parameters. We propose a number of novel algorithms for performing the drafting phase. Each algorithm is based on the computation of a parameter usually defined in relation to association rules: leverage, conviction, lift, Pearson's $X^2$ or Cramer Index.

We experimented the algorithms on four different Bayesian networks: “Visit to Asia”, “Car_diagnosis”, “Alarm” and "Boelarge": the algorithm BNL-rules with $X^2$ obtained the best results and performed better that SLA in two cases and equally well in two other two cases better. By starting the second and the third learning phase of SLA from a better structure it is possible to achieve a better final result.

In the future we plan to compare SLA and BNL-rules on larger networks in order to obtain a more definitive confirmation of the validity of the approach.

## References

[1] Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Buneman, P., Jajodia, S. (eds.): Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993. ACM Press 22(2) (1993) 207-216

[2] Akaike, H.: A new Look at Statistical Model Identification. IEEE Trans. Automatic Control 19 (1974) 716-723

[3] Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. In: Hunter, J. (ed.): Proceedings of the Second European Conference on Artificial Intelligence in Medicine (AIME 89). Springer, Berlin (1989) 247-256

[4] Berry, J.A., Linoff, G.S.: Data Mining Techniques for Marketing, Sales and Customer Support. John Wiley & Sons Inc., New York (1997)

[5] Boerlage, Brent: Link Strength in Bayesian Networks. MSc Thesis, Dept. Computer Science, Univ. of British Columbia, BC (1992).

[6] Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic Itemset Counting and Implication Rules for Market Basket Data. In: Peckham, J. (ed.): Proceedings ACM SIGMOD International Conference on Management of Data. SIGMOD 26(2) (1997) 255-264

[7] Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: Generalizing association rule to correlations. In: Proceedings ACM SIGMOD International Conference on Management of Data. SIGMOD (1997) 265-276

[8] Cheng, J., Greiner, R., Kelly, J., Bell, D., Liu, W.: Learning Bayesian networks from data: An information-theory based approach, Artificial Intelligence 137 (2002) 43–90
[9] Chow, C.K., Liu, C.N.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory 14 (1968) 462-467
[10] Cooper, G., Herskovits, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9 (1992) 309-347
[11] Giudici, P.: Data mining - Metodi statistici per le applicazioni aziendali. McGraw-Hill, Milano (2001)
[12] Heckerman, D., Geiger, D., Chickering, D.: Learning Bayesian Networks: the combination of knowledge and statistical data. Machine Learning 20 (1995) 197-243
[13] Heckerman, D.: Tutorial on learning in Bayesian networks. In: Jordan, M. (ed.): Learning in Graphical Models. MIT Press, Cambridge, MA (1999)
[14] Herskovits, E.H.: Computer-based probabilistic-network construction. Doctoral Dissertation, Medical Informatics, Stanford University (1991)
[15] Hugin, http://www.hugin.com
[16] Lauritzen, S.L., Spiegelhalter D.J.: Local computations with probabilities on graphical structures and their application to expert systems. J. Royal Statistics Society B 50(2) (1988) 157-194
[17] Madigan, D., Raftery A.:. Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window. J. Am. Statist. Association 89 (1994) 1535-1546
[18] Netica, http://www.norsys.com
[19] Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann, San Francisco (1988)
[20] Rissanen, J.: Stochastic Complexity (with discussion). J. Roy. Statist. Soc. B 49 (1987) 223-239
[21] Sigh, M., Valtorta, M.: Construction of Bayesian Network Structures from Data: a Brief Survey and an Efficient Algorithm. In: Heckerman, D., Mamdani, A. (eds.): Proceedings of the Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco (1993) 259–265
[22] Silverstein, C., Brin, S., Motwani, R., Ullman, J.D.: Scalable Techniques for Mining Causal Structures. Data Mining and Knowledge Discovery 4(2/3) (2000) 163-192
[23] Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search. Lecture Notes in Statistics. Springer, Berlin (1993)
[24] Spirtes, P., Glymour, C., Scheines, R.: An algorithm for fast recovery of sparse causal graphs. Social Science Computer Review 9 (1991) 62-72
[25] Suzuki, J.: Learning Bayesian Belief Networks Based on the MDL principle: An Efficient Algorithm Using the Branch and Bound Technique. IEICE Transactions on Communications Electronics Information and Systems (1999)
[26] WEKA http://www.cs.waikato.ac.nz/~ml/weka/doc_gui /weka.associations.ItemSet. html#leverageForRule

# Incremental Induction of Rules for Document Image Understanding

Stefano Ferilli, Nicola Di Mauro, Teresa M.A. Basile, and Floriana Esposito

Dipartimento di Informatica
Università di Bari
via E. Orabona, 4 – 70125 Bari – Italia
{ferilli,basile,nicodimauro,esposito}@di.uniba.it

**Abstract.** This paper aims at presenting the application of first-order logic machine learning techniques to two document domains in order to learn rules for recognizing the semantic role of their logical components. Specifically, the multistrategy incremental learning system INTHELEX has been applied to multi-format scientific papers and documents concerning European films from the 20's and 30's. The challenge comes from the different levels of formatting standards in these domains: from (more or less) standardized layouts, in scientific papers, to documents with almost no standard, in historical cultural heritage material. Experimental results in both domains and a comparison with the Progol system assess the advantages that the exploitation of INTHELEX can yield.

## 1 Introduction

One of the key issues regarding the use of information systems is the acquisition of new information, which often resides in paper documents. Automatically acquiring new information could be useful to populate knowledge bases to be exploited in support of human activities in many tasks. This holds, in particular, when dealing with important historic and cultural sources, which are fragile and distributed in various archives, and must be digitized in order to preserve them and make them available to a wider public. If such document bases are to be exploited by human experts, which happens for instance in the cultural heritage domain, the need of having learned rules expressed in such a way that they can be easily understood and checked is particularly strong.

In this context, automatic induction of rules that are able to recognize the semantic role of the significant layout components of documents in order to provide them to the domain experts would be very helpful in order to make the extraction of important information easier. In particular, the complexity of some document layout structures suggests the use of symbolic (first-order logic) descriptions and techniques. This was the motivation that led us to try applying the learning system INTHELEX in this environment. Incremental learning is necessary when incomplete information is available at the time of initial theory generation, which is very frequent in real-world situations like those described above, where a continuous flow of new documents requires a constant check

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 176–188, 2003.
© Springer-Verlag Berlin Heidelberg 2003

(and possible revisions) of the learned theories. Hence, the need for incremental algorithms to complete and support the batch ones, that perform learning in one step and thus require the whole set of observations to be available from the beginning. The corpus of documents on which we worked concerns two domains: multi-format scientific papers and rare historic film censorship documents from the 20's and 30's. All material is analyzed, indexed, annotated and interlinked by domain experts, to which it could be useful to provide suitable knowledge management tools to support their work.

The following section presents the incremental learning system along with its reasoning strategies; then, Section 3 reports the experimental results obtained on the scientific papers and cultural heritage collections and the comparison of INTHELEX with Progol [8]. Lastly, Section 4 draws some conclusions.

## 2 Incremental Learning with INTHELEX

Automatic revision of logic theories is a complex and computationally expensive task. Considerations on the incremental learning systems available in the literature [4] led to the design and implementation of INTHELEX (INcremental THEory Learner from EXamples), whose most characterizing features are its incremental nature, the reduced need of a deep background knowledge, the exploitation of negative information and the peculiar bias on the generalization model, which reduces the search space and does not limit the expressive power of the adopted representation language.

### 2.1 The Inductive Core

INTHELEX is a learning system for the induction of *hierarchical* logic theories from examples [4]: it is *fully incremental* (in addition to the possibility of refining a previously generated version of the theory, learning can also start from an empty theory); it is based on the *Object Identity* assumption (terms, even variables, denoted by different names within a formula must refer to different objects) and learns theories expressed as sets of Datalog$^{OI}$ clauses [10] from positive and negative examples; it can learn simultaneously *multiple concepts*, possibly related to each other (recursion is not allowed); it retains all the processed examples, in order to guarantee validity of the learned theories on all of them; it is a *closed loop* learning system (i.e. a system in which feedback on performance is used to activate the theory revision phase [1]).

INTHELEX incorporates two inductive refinement operators, one for generalizing hypotheses that reject positive examples (upward), and the other for specializing hypotheses that explain negative examples (downward). It exploits a (possibly empty) previous theory, a graph describing the dependence relationships among concepts (if any), and a historical memory of all the past examples that led to the current theory. Whenever a new example is taken into account, it is stored in such a repository and the current theory is checked against it.

The generalization process is started if a positive example is not explained. Starting from the current theory, the misclassified example and the past negative examples, it ends with a revised theory. Since there is no single clause responsible for not explaining the given example, the system must firstly choose a clause to be generalized among those making up the involved concept description. Then, the system tries to compute one of the $lgg_{OI}$'s of this clause and the example. The upward refinement operator extends the concept of *least general generalization* (*lgg*) introduced by Plotkin [9] to cope with the ordering induced by $\theta_{OI}$-subsumption. Note that the *lgg* is no longer unique under $\theta_{OI}$-subsumption, since the space of Datalog clauses is not a lattice when ordered by $\theta_{OI}$-subsumption [10], while it is when ordered by $\theta$-subsumption. In our framework, the $lgg_{OI}$ is initialized as the literal generalizing the two heads; then, literals in Plotkin's *lgg* are progressively added to the partial generalization as long as they share at least one variable with it and fulfill the $OI$ requirement. When a generalization is found, it is tested with respect to all the past negative examples. If none of them is explained, then the generalized clause is replaced in the theory by the generalization. Otherwise, an intelligent backtracking on the literals in Plotkin's *lgg* is performed, which yields another generalization (if any). If no clause in an incomplete definition can be generalized so that the resulting theory is complete and consistent, the system checks if the example itself, with the constants properly turned into variables, is consistent with the past negative examples. Such a clause is added to the theory, or else the example itself is added as an exception to the theory.

As to specialization, in our framework it consists of adding proper literals to an inconsistent clause, in order to avoid it explaining a negative example. Revisions performed by our operator are always minimal [11]: a specialization obtained by turning a variable into a constant is not provided since all clauses in the theory contain only variables as arguments. Starting from the current theory, the misclassified example and the past positive examples, the specialization algorithm yields a revised theory by adding proper literals to the inconsistent clauses. Since the space in which the literals to be added should be searched for is potentially infinite, the operator has to focus the search into the portion of the space that contains the solution of the diagnosed error. The search is firstly performed in the space of positive literals, that contains information coming from all the past positive examples, but not yet exploited by it. If the search in this space fails, the algorithm autonomously performs a *representation change*, extending the search to the space of negative literals, built by considering the negative example that caused the error [10]. First of all, the process detects all the clauses occurring in the SLD-derivation of the example. Then, the system tries to specialize one at the lowest possible level (which corresponds to a clause defining a concept whose level is the lowest in the dependency graph), in order to refine more "basic" concepts first. Since the downward refinements we are looking for must satisfy the property of maximal generality, this suggests to search for one (or more) positive literal(s) which can discriminate all the past positive examples from the current negative one. Specifically, if there exists one

(or a combination of) literal(s) that, when added to the body of the clause to be specialized, is able to discriminate the negative example that caused the inconsistency, then the downward refinement operator should be able to find it. If the derivation of an example in a theory is not unique, a single specialization step could be insufficient to restore the consistency of the theory. For such a reason, when a specialization of a clause is reached, the example is tested again to check if another SLD-derivation, different from the previous one, exists for it. If so, the process is iterated until no more derivations of the example are found. Note that only one step of specialization is needed whenever the SLD-derivation of the example involves just one clause. If no clause in the SLD-derivation of the example can be specialized by adding literals coming from the positive examples, an attempt is made to add a negative literal to the first clause of the SLD-derivation (the one related to the concept the example is an instance of). If none of the clauses obtained makes the theory complete and consistent again, then the system adds the negative example to the theory as an exception.

### 2.2 Multistrategy Learning

Another peculiarity in INTHELEX is the integration of multistrategy operators that may help in the solution of the theory revision problem by pre-processing the incoming information, according to the theoretical framework for integrating different learning strategies known as Inferential Learning Theory [7]. Namely, deduction is exploited to fill observations with information that is not explicitly stated, but is implicit in their description. Conversely, abduction aims at completing possibly partial information in the examples (adding more details), whereas abstraction removes superfluous details from the description of both the examples and the theory. Thus, even if with opposite perspectives, both aim at reducing the computational effort required to learn a correct theory with respect to the incoming examples.

INTHELEX requires the observations to be expressed only in terms of the set of predicates that make up the description language for the given learning problem. To ensure uniformity of the example descriptions, such predicates have no definition. Nevertheless, since the system is able to handle a hierarchy of concepts, combinations of these predicates might identify higher level concepts that is worth adding to the descriptions in order to raise their semantic level. For this reason, the system implements a saturation operator that exploits deduction to recognize such concepts and explicitly add them to the examples description. Any time a new example is considered, a preliminary saturation phase can be performed, that adds the higher level concepts whose presence can be deduced from such rules by subsumption and/or resolution. Note that all the specific information used by saturation is left in the example description and it is preserved in the learning process until other evidence reveals it is not significant for the concept definition.

Abduction was defined by Peirce as hypothesizing some facts that, together with a given theory, could explain a given observation. According to the framework proposed in [5], an *abductive logic theory* is made up by a normal logic

program [6], a set of *abducibles* and a set of *integrity constraints*. Abducibles are the predicates about which assumptions (*abductions*) can be made: They carry all the incompleteness of the domain (if it were possible to complete these predicates then the theory would be correctly described). Integrity constraints (each corresponding to a combination of literals that is not allowed to occur) provide indirect information about them. The proof procedure implemented in the system starts from a goal and a set of initial assumptions and results in a set of consistent hypotheses (abduced literals) by intertwining *abductive* and *consistency derivations*. Intuitively, an abductive derivation is the standard Logic Programming derivation suitably extended in order to consider abducibles [4].

The exploitation of abstraction concerns the shift from the language in which the theory is described to a higher level one. According to the framework proposed in [12], concept representation deals with entities belonging to three different levels. Concrete objects reside in the *world*, but any observer's access to it is mediated by his *perception* of it. To be available over time, these stimuli must be memorized in an organized *structure*, an *extensional* representation of the perceived world. Finally, to reason about the perceived world and communicate with other agents, a *language* is needed, that describes it *intensionally*. Abstraction takes place at the world-perception level by means of a set of operators, and then propagates to higher levels, where it is possible to identify operators corresponding to the previous ones. An abstraction theory expresses such operators, that allow the system to replace a number of components by a compound object, to decrease the granularity of a set of values, to ignore whole objects or just part of them, and to neglect the number of occurrences of some kind of object. The abstraction theory must be provided to the system that automatically applies it to the learning problem at hand before processing the examples.

## 3 Experimental Results

INTHELEX has been exploited to learn rules for the automatic identification of logical components in two documents domains which are characterized by different (if any at all) formatting style in their layout structure. The two datasets consist of scientific papers belonging to three different series (*ICML*, *ISMIS*, *IEEET*) and of a large collection of documents concerning film censorship, used in the European Project COLLATE, coming from three historical European film archives (*FAA*, *DIF*, *NFA*). In the former, each series is characterized by its own layout standard, but fulfilled more or less strictly in the three cases. In the latter, a challenge comes from the low layout quality and different formatting standard of the material, which causes a considerable amount of noise in its description. In this environment, for example, abstraction can be exploited for ignoring superfluous specks on the layout, while abduction could be useful to hypothesize the presence of expected but unseen layout components.

The first-order descriptions of such documents, needed to run INTHELEX, were automatically generated by the system WISDOM++ [3] (note that the scientific papers dataset is the same as in [3], but the set of learned layout compo-

```
width_very_small(X) :- width(X,Y), Y >= 0, Y =< 15.
width_small(X) :- width(X,Y), Y > 15, Y =< 30.
width_medium_small(X) :- width(X,Y), Y > 30, Y =< 80.
width_medium(X) :- width(X,Y), Y > 80, Y =< 130.
width_medium_large(X) :- width(X,Y), Y > 130, Y =< 250.
width_large(X) :- width(X,Y), Y > 250, Y =< 400.
width_very_large(X) :- width(X,Y), Y > 400, Y =< 640.
```

**Fig. 1.** Abstraction rules for width of block

nents for each of the classes is different). Starting from scanned images, it is able to identify the layout blocks that make up a paper document and to describe them in terms of their size (height and width, in pixels), position (horizontal and vertical, in pixels from the top-left corner), type (text, line, picture and mixed) and relative position (horizontal/vertical alignment, adjacency). Since the inductive procedure embedded in INTHELEX is not able to handle numeric values (such as the number of pixels in the document descriptions provided by WISDOM++), a change of representation in the description language was necessary, such that final observations were made up of symbolic attributes only. The abstraction operator was used for breaking numeric values into intervals represented by symbolic constants, by providing the system with an abstraction theory containing rules that encode such a language shift. Figure 1 shows the rules of the abstraction theory that are in charge of discretizing the width of a layout block in a document. All the experiments described in the following were performed according a 10-fold cross validation methodology. For each class of documents in these datasets, the layout blocks that are semantically significant for indexing/retrieval purposes were identified and annotated by expert users, and subsequently used as examples to learn rules for automatically recognize them when new documents become available. Note that different document classes have different labels, as reported in the following (in square brackets the corresponding number of positive and negative instances are reported).

Let us start from the scientific papers dataset. The semantic labels of interest recognized by the domain experts for class *ICML* were: *abstract* [28+,340-], *author* [36+, 332-], *page_number* [27+, 341-] and *title* [29+, 339-]. As regards class *ISMIS* the following labels characterizing the objects belonging to it were provided: *abstract* [32+, 250-], *affiliation* [30+, 252-], *author* [30+, 252-] and *title* [30+, 252-]. Finally, for class *IEEET* we had: *abstract* [36+, 576-], *affiliation* [31+, 581-], *author* [34+, 578-], *index_term* [32+, 580-], *title* [40+, 572-], *running_head* [40+, 572-] and *page_number* [33+, 579-]. Each positive example for a label class to be learned was considered as negative for the others. Furthermore, any document block not labelled by the expert as significant was considered negative for all the components to be learned. Table 1, discussed at the end of this section, reports the experimental results averaged on the 10 folds as regards number of clauses defining the concept ($Cl$), number of performed generalizations ($Lgg$), *Runtime* (in seconds) and Predictive Accuracy.

**Table 1.** Statistics for understanding in scientific papers domain

| | Clauses | Lgg's | Runtime | Accuracy % | | |
|---|---|---|---|---|---|---|
| ICML | | | | INTHELEX | PROGOL | $t$ VALUE |
| *abstract* | 2.00 | 7.70 | 66.61 | 98.64 | 99.45 | 1.96 |
| *author* | 1.10 | 5.50 | 47.77 | 99.73 | 98.09 | -2.73 |
| *page_number* | 2.70 | 6.40 | 233.65 | 98.65 | 97.26 | -1.48 |
| *title* | 3.30 | 12.90 | 515.14 | 98.08 | 97.79 | -0.38 |
| ISMIS | | | | | | |
| *abstract* | 4.50 | 12.30 | 92.31 | 96.44 | 95.02 | -0.87 |
| *affiliation* | 5.30 | 13.30 | 515.61 | 94.65 | 91.81 | -1.30 |
| *author* | 4.70 | 12.80 | 857.26 | 95.02 | 91.13 | -1.99 |
| *title* | 3.80 | 11.40 | 329.41 | 95.75 | 96.84 | 0.78 |
| IEEET | | | | | | |
| *abstract* | 5.90 | 18.20 | 594.63 | 96.89 | 97.87 | 1.62 |
| *affiliation* | 3.60 | 10.40 | 331.38 | 98.53 | 98.85 | 0.68 |
| *author* | 2.20 | 10.50 | 195.28 | 99.18 | 98.36 | -1.85 |
| *index_term* | 4.20 | 13.90 | 892.92 | 97.88 | 98.53 | 1.18 |
| *page_number* | 3.80 | 13.90 | 647.71 | 97.38 | 99.18 | 3.16 |
| *running_head* | 5.10 | 15.90 | 766.62 | 95.91 | 97.87 | 2.71 |
| *title* | 3.50 | 12.70 | 474.93 | 97.39 | 95.42 | -2.35 |

Figure 2 shows the definitions learned by INTHELEX for the target concepts of the *ICML* set of documents in one of the 10 folds. As shown in Figure 3, it is possible to exactly recognize and map on a sample document the layout blocks referred to in the rules. For instance, the rule describing the label *abstract* says that a block $A$ is an abstract of a document $B$ belonging to *ICML* collection if its width is medium-large and it is placed on the left (w.r.t. the horizontal position) middle (w.r.t. the vertical position) part of the document. This definition also refers to other three objects, $C$, $D$ and $E$, one of which, $C$, has smaller size than the block $A$ and is placed above $A$. The domain expert recognized block $C$ as that containing the title (word) "Abstract" in a paper. Note that starting from document descriptions whose length ranges between 100 and 180 literals, the learned rules, shown in Figure 2, contain at most 15 literals.

In the COLLATE domain, the document layout quality is often affected by manual annotations, stamps that overlap to sensible components, ink specks, etc. As to the layout standard, many documents are typewritten sheets, that consist of all equally spaced lines in Gothic type. Such a situation requires the system to be flexible in the absence of particular layout components due to the typist's style, and to be able to ignore layout details that are meaningless or superfluous to the identification of the interesting ones. Note that the symbolic method adopted allows the trainer to specifically select prototypical examples to be included in the learning set. This explains why theories with good predictiveness can be obtained even from fewer observations.

```
logic_type_abstract(A) :-
   part_of(B,A),width_medium_large(A),pos_left(A),pos_middle(A),
   part_of(B,C), part_of(B,D),part_of(B,E),on_top(C,A),
   width_medium_small(C),height_very_very_small(C),type_of_text(C).
logic_type_author(A) :-
   part_of(B,A),type_of_text(A),pos_upper(A),
   part_of(B,C),height_very_very_small(C),type_of_text(C),pos_upper(C),
   part_of(B,D),on_top(D,A),on_top(E,D),pos_center(E),pos_upper(E),
   width_very_large(D),height_smallest(D),pos_center(D),pos_upper(D),
logic_type_page_number(A) :-
   part_of(B,A),pos_upper(A),part_of(B,C),
   width_very_large(C),height_smallest(C),pos_center(C),pos_upper(C),
   on_top(A,E),width_very_large(E),pos_center(E),pos_upper(E),
   on_top(D,C),type_of_text(D),pos_center(D),pos_upper(D).
logic_type_title(A) :-
   part_of(B,A),type_of_text(A),pos_center(A),pos_upper(A),
   part_of(B,C),height_very_very_small(C),type_of_text(C),
   on_top(C,D),on_top(A,E),part_of(B,E),
   width_very_large(E),height_smallest(E),pos_center(E),pos_upper(E).
```

**Fig. 2.** Learned definition for semantic components of ICML

As for the previous dataset, each document in this domain has different semantic components according the class it belongs to. Thus, in the documents coming from the *DIF* archive, which are censorship decisions, the domain experts recognized the following characterizing labels: *cens_signature* [35+], *cert_signature* [35+], *object_title* [36+], *cens_authority* [36+], *chairman* [36+], *assessors* [36+], *session_data* [36+], *representative* [49+]. Definitions for each class were learned, starting from the empty theory and considering as negative examples for each label those that were positive for the other components. Table 2 reports the experimental results, averaged on the 10 folds, of the interpretation process in this environment.

**Table 2.** Statistics for understanding of DIF documents

| | Clauses | Lgg's | Runtime | Accuracy % | | |
|---|---|---|---|---|---|---|
| | | | | INTHELEX | PROGOL | $t$ VALUE |
| *cens_signature* | 2.2 | 11.6 | 1459.883 | 98.32 | 99.33 | 1.41 |
| *cert_signature* | 2.2 | 7.6 | 176.592 | 98.31 | 99.32 | 1.96 |
| *object_title* | 5 | 15.2 | 3960.829 | 94.66 | 94.67 | 0.01 |
| *cens_authority* | 2.9 | 12.1 | 2519.45 | 97.64 | 98.02 | 0.30 |
| *chairman* | 4.6 | 13.8 | 9332.845 | 93.10 | 91.97 | -0.92 |
| *assessors* | 4.6 | 15 | 12170.93 | 94.48 | 98.52 | **4.71** |
| *session_data* | 2.5 | 8.6 | 1037.96 | 97.68 | 98.98 | 1.76 |
| *representative* | 5.6 | 20.7 | 13761.958 | 92.98 | 96.98 | 2.88 |

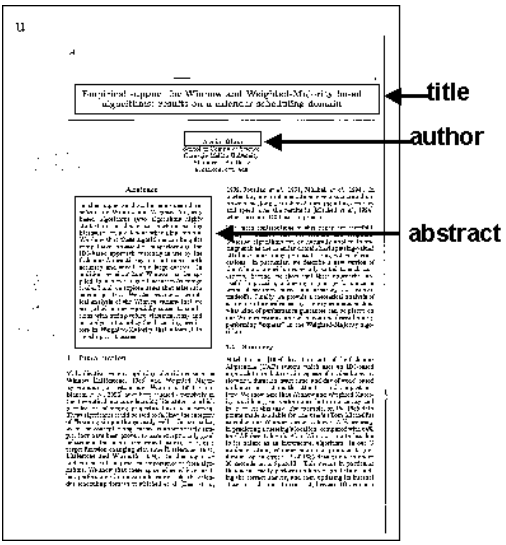



**Fig. 3.** Sample ICML document

As regards the class of documents coming from the *FAA* archive, the provided labels were: *registration_au* [28+], *date_place* [26+], *department* [17+], *applicant* [11+], *reg_number* [28+] , *film_genre* [20+], *film_length* [19+], *film_producer* [18+], *film_title* [20+]. Averaged results are reported in Table 3. Figure 4 shows the definition learned by INTHELEX for the concept *film_title* of the registration cards from *FAA* in one of the folds. Such a rule describes the identified blocks by means of their size and relative position w.r.t. the whole document and the other blocks. Figure 5 graphically highlights the components of the learned description in a sample of document. Also in this case, as for the scientific papers,

**Table 3.** Statistics for understanding of FAA documents

| | Clauses | Lgg's | Runtime | Accuracy % | | |
|---|---|---|---|---|---|---|
| | | | | INTHELEX | PROGOL | $t$ VALUE |
| *registration_au* | 5.6 | 12.5 | 3739.366 | 91.43 | 89.35 | -1.28 |
| *date_place* | 6.9 | 13.5 | 7239.625 | 86.69 | 89.80 | 1.94 |
| *department* | 1.9 | 6.6 | 118.625 | 98.95 | 100 | 1.00 |
| *applicant* | 2 | 4.5 | 93.993 | 97.89 | 98.91 | 1.49 |
| *reg_number* | 5.1 | 14.4 | 4578.208 | 91.95 | 97.30 | **4.82** |
| *film_genre* | 4 | 8.4 | 2344.899 | 93.02 | 94.18 | 0.38 |
| *film_length* | 5.5 | 9.9 | 3855.391 | 90.87 | - | - |
| *film_producer* | 4.9 | 10.4 | 4717.17 | 94.03 | - | - |
| *film_title* | 5.4 | 11.1 | 4863.084 | 89.82 | - | - |

```
logic_type_film_title(A) :-
   page_first(B),part_of(B,A),part_of(B,C),part_of(B,D),
   type_of_text(A),pos_upper(A),type_of_text(D),pos_upper(D),
   height_very_small(C),type_of_text(C),pos_center(C),pos_upper(C),
   width_very_large(E),height_smallest(E),
   type_of_hor_line(E),pos_center(E),pos_upper(E),
   height_very_very_small(F),pos_left(F),pos_upper(F),
   on_top(E,D),on_top(E,F),alignment_left_col(F,A).
```

**Fig. 4.** Learned definition for a semantic component of registration cards in FAA

starting from document descriptions whose length ranges between 40 and 379 literals, the learned rule, shown in Figure 4, contains 23 literals.

Finally, documents belonging to *NFA* archive were characterized by the following labels, almost all different from the previous ones: *dispatch_office* [33+], *applic_notes* [18+], *no_censor_card* [21+], *film_producer* [20+], *no_prec_doc* [20+], *applicant* [22+], *film_genre* [17+], *registration_au* [25+], *cens_process* [30+], *delivery_date* [16+] and *cens_card* [26+]. Table 4 shows the averaged results.

The overall outcomes reveal that INTHELEX was actually able to learn significant definitions for the layout blocks of interest in both the more standard-formatted documents and the low-quality ones. Indeed, the predictive accuracy is always very high in both cases, reaching even 99.73% in the case of scientific documents (in which never falls below 94.65%) and 98.95% in the COLLATE dataset (in which only 2 cases out of 28 fall below 90%, with percentages 86.69% and 89.82%). As shown, the learned rules have a high degree of understandability for human experts, which was one of the requirements for the experiment. As expected, the interpretation problem, that is concerned with the semantics of the layout blocks inside documents, turned out to be easier in the standard-formatted documents (scientific papers) than in the historical material. This is suggested by the tendential increase in runtime from Table 1 to Tables 2, 3 and 4. Such an increase is in some cases particularly evident, but it should be considered

**Fig. 5.** Components in the learned definition of a FAA registration card B

**Table 4.** Statistics for understanding of NFA documents

| | Clauses | Lgg's | Runtime | Accuracy % | | |
|---|---|---|---|---|---|---|
| | | | | INTHELEX | PROGOL | $t$ VALUE |
| *dispatch_office* | 6.8 | 13.9 | 13149.31 | 94.28 | 94.86 | 0.55 |
| *applic_notes* | 2.5 | 5.7 | 231.05 | 98.81 | 100 | 1.82 |
| *no_censor_card* | 5.3 | 11.2 | 8136.796 | 95.48 | 96.38 | 0.89 |
| *film_producer* | 4.9 | 9.6 | 5303.78 | 93.98 | 97.29 | 2.29 |
| *no_prec_doc* | 4.6 | 11 | 5561.14 | 93.97 | 94.59 | 1.02 |
| *applicant* | 6.7 | 11.5 | 15588.15 | 93.66 | 93.38 | -0.26 |
| *film_genre* | 2.8 | 6.9 | 684.35 | 98.53 | 99.41 | 1.96 |
| *registration_au* | 4.1 | 12.5 | 5159.74 | 94.64 | 97.31 | 2.23 |
| *cens_process* | 4.8 | 10.8 | 4027.90 | 98.51 | - | - |
| *cens_card* | 5.6 | 11.8 | 3363.86 | 94.61 | 95.23 | 0.72 |
| *delivery_date* | 4 | 9.1 | 3827.34 | 95.51 | 98.82 | **4.31** |

that the high predictive accuracy should ensure that few theory revisions can be expected when processing further documents.

The performance of the system on these datasets were compared to that obtained by the Progol batch system. For pairwise comparison a 10-fold cross validation paired $t$-test was used [2] in order to evaluate the difference in effectiveness of the rules induced by the two systems according to the predictive accuracy metric. Requiring a significance level of $\alpha = 0.995$, the test revealed no statistically significant differences in predictive accuracy among them in the scientific papers domain, while there are in the COLLATE dataset in favor of Progol for the following layout components: *assessors* for documents belonging to DIF dataset (Table 2), *reg_number* for FAA (Table 3) and *delivery_date* for NFA (Table 4). Such a result was expected in this domain because of the noise affecting historical material, given the approaches followed by the two systems. Indeed, batch systems may take into account the whole knowledge available since the beginning of the learning process, so having a general vision of the knowledge available, while incremental systems must work having, at any moment, only a limited vision of the domain.

However, it is worth noting that Progol was not able to learn some layout components at all. This happened for *film_length*, *film_title* and *film_producer* from FAA (where Progol learned theories made up of only positive exceptions, thus lacking any predicive accuracy) and *cens_process* of NFA (where after two days the process was stopped because no theory had been induced yet, for none of the folds). Such a difficulty in learning some components for Progol is confirmed by the presence of many positive exceptions also in the other learned theories (in some cases an average of 16 exceptions on 27 examples of the target concept are present). The failure in learning a theory, even for a single concept, is unacceptable in real-world applications like the one under consideration. Thus, such a comparison turns out encouraging on the possibility of exploiting INTHELEX in difficult domains, not only when the observations are provided incrementally.

## 4 Conclusions and Future Work

This paper presented experimental results proving that the incremental learning system INTHELEX can be successfully exploited to learn rules for automatic interpretation of scientific and cultural heritage documents. This was confirmed by a comparison with the batch learning system Progol on the predictive accuracy metric, revealing that there are generally no statistically significant differences among the two systems. A few situations are in favor of Progol, but its complete failure in learning some concepts is unacceptable in real-world applications. Thus, future work will concern deeper exploitation of the INTHELEX features aimed at improving its performance.

## Acknowledgements

This work was partially funded by the EU project IST-1999-20882 COLLATE.

## References

[1] J. M. Becker. Inductive learning of decision rules with exceptions: Methodology and experimentation. B.s. diss., Dept. of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1985. UIUCDCS-F-85-945. 177

[2] Thomas G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998. 186

[3] F. Esposito, D. Malerba, and F. A. Lisi. Machine learning for intelligent processing of printed documents. *Journal of Intelligent Information Systems*, 14(2/3):175–198, 2000. 180

[4] F. Esposito, G. Semeraro, N. Fanizzi, and S. Ferilli. Multistrategy Theory Revision: Induction and abduction in INTHELEX. *Machine Learning Journal*, 38(1/2):133–156, 2000. 177, 180

[5] E. Lamma, P. Mello, F. Riguzzi, F. Esposito, S. Ferilli, and G. Semeraro. Cooperation of abduction and induction in logic programming. In A. C. Kakas and P. Flach, editors, *Abductive and Inductive Reasoning: Essays on their Relation and Integration*. Kluwer, 2000. 179

[6] J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, second edition, 1987. 180

[7] R. S. Michalski. Inferential theory of learning. developing foundations for multistrategy learning. In R. S. Michalski and G. Tecuci, editors, *Machine Learning. A Multistrategy Approach*, volume IV, pages 3–61. Morgan Kaufmann, San Mateo, CA, U. S. A., 1994. 179

[8] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995. 177

[9] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970. 178

[10] G. Semeraro, F. Esposito, D. Malerba, N. Fanizzi, and S. Ferilli. A logic framework for the incremental inductive synthesis of datalog theories. In N. E. Fuchs, editor, *Logic Program Synthesis and Transformation*, number 1463 in Lecture Notes in Computer Science, pages 300–321. Springer-Verlag, 1998. 177, 178

[11] S. Wrobel. *Concept Formation and Knowledge Revision*. Kluwer Academic Publishers, 1994. 178

[12] J.-D. Zucker. Semantic abstraction for concept representation and learning. In R. S. Michalski and L. Saitta, editors, *Proceedings of the 4th International Workshop on Multistrategy Learning*, Desenzano del Garda, Italy, 1998. 180

# Explaining Bagging with Monte Carlo Theory

Roberto Esposito[1] and Lorenza Saitta[2]

[1] Università di Torino
Dipartimento di Informatica
Corso Svizzera 185, 10149 Torino, Italy
esposito@di.unito.it
[2] Università del Piemonte Orientale
Dipartimento di Informatica,
Spalto Marengo 33, 15100 Alessandria, Italy
saitta@mfn.unipmn.it

**Abstract.** In this paper we propose the use of the framework of Monte Carlo stochastic algorithms to analyze ensemble learning, specifically, *bagging*. In particular, this framework allows one to explain bagging's behavior and also why increasing the margin improves performances. Experimental results support the theoretical analysis.

## 1 Introduction

Ensemble learning has been brought to the attention of the Machine Learning (ML) community by Schapire [17], who suggested that combining several learned hypotheses instead of choosing one could be a winning strategy in classification/regression tasks. Since then, ensemble learning has been actively investigated [11, 12, 3, 19].

Advantages of ensemble learning have been discussed by Dietterich [5] in a recent paper. Among others, the amazing effectiveness and robustness to overfitting are responsible for attracting a great deal of interest. In fact, it often happens that the generalization error still decreases after the training error reaches zero [16, 6]. In the attempt to explain this appealing behavior, many theoretical and empirical works have tried to relate ensemble learning to results from other fields, such as Game theory [10, 12, 18], Logistic regression [13, 4], and Convex optimization [15].

In this paper we limit ourselves to a specific ensemble learning algorithm, i.e., *Bagging* [3], and we propose to analyze its properties from the point of view of Monte Carlo algorithms theory. This new parallel sheds some light on the foundations of bagging. In particular:

- it suggests the conditions under which bagging is likely to be better than single hypothesis learning,
- it explains why margin reduction increases performances, and
- it allows some reasons for robustness to overfitting to be conjectured.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 189–200, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Bagging [3] is one among the most studied ensemble learning algorithms. It takes in input a learning set $\mathbf{L}_0$, an integer number $T$, and a (possibly weak) induction algorithm $\mathcal{A}$. At each iteration $t$ the algorithm creates a bootstrap [7] replicate $\mathbf{L}$ of the original learning set, and uses it as input to $\mathcal{A}$ to form a new hypothesis $h_t$ . When $T$ hypotheses have been learned, they are combined using a simple majority voting rule.

Experimental analysis of bagging's performances is provided by Bauer and Kohavi [1], Quinlan [14] and Dietterich [5].

Breiman [3] found a significant increase of the classification accuracy, when the hypotheses are bagged, on different datasets from the UCI repository, for both regression and classification. Moreover, in the same paper, a theoretical analysis is carried on. The notion of *order correct* classifier is introduced, and it is shown that when $\mathcal{A}$ is order correct, then bagging is expected to behave almost optimally (i.e., it behaves like the Bayes classifier). Throughout the paper it is claimed that "bagging stable classifiers is not a good idea". A "stable" classifier is qualitatively defined as one that does not show large variations in the learned hypotheses when the training set undergoes small variations.

## 2 Monte Carlo Algorithms

In this section we recall some basic notions from Monte Carlo algorithms theory. In the literature, the name "Monte Carlo" often denotes a generic stochastic algorithm; in this paper we will refer to the more restrictive definition given by Brassard and Bratley [2].

**Definition 1.** *Let $\Pi$ be a class of problems, $\mathcal{Y}$ a finite set of answers for the problems in $\Pi$, and $c : \Pi \rightarrow 2^{\mathcal{Y}}$ be a function that maps each problem into the set of its correct answers. A stochastic algorithm $\mathcal{A}$ mapping $\Pi$ into $\mathcal{Y}$ is said to be a Monte Carlo algorithm if it always terminates, when applied to an instance $x \in \Pi$, providing an answer to $x$, which may occasionally be incorrect.*

The two following properties of Monte Carlo algorithms turn out to be relevant to the goals of this paper.

**Definition 2.** *A Monte Carlo algorithm is* consistent *if it never outputs two distinct correct answers for the same problem $x$.*

We may notice that, whenever $c(x)$ reduces to a mapping from $\Pi$ to $\mathcal{Y}$ (instead of $2^{\mathcal{Y}}$), the algorithm is consistent.

**Definition 3.** *A Monte Carlo algorithm is* p-correct *if the probability that it gives a correct answer to $x$ is at least $p$, independently of the specific problem instance $x$ considered.*

**Definition 4.** *Given a $p$-correct Monte Carlo algorithm, the difference $\gamma = p - 1/2$ is called the* advantage *of the algorithm.*

The great interest in Monte Carlo algorithms resides in their *amplification* ability.

**Theorem 1.** (Brassard and Bratley [2]) *Let $\Pi$ be a class of problems, and let MC be a Monte Carlo algorithm both consistent and p-correct over $\Pi$. By running MC $T$ times on any instance $x \in \Pi$ and taking the majority answer as result, the probability that $x$ is correctly classified increases exponentially with $T$, provided that $p$ is greater than* $1/2$.

In the next section we will apply the notions introduced so far to learning.

## 3 A Monte Carlo Ensemble Learner

For the sake of simplicity, we only consider, in this paper, binary classification problems, i.e., $\mathcal{Y} = \{+1, -1\}$. Let $\mathcal{X}$ be a finite set of examples[1], $c$ a target concept, and $\mathcal{A}$ a (weak) learner. Elements of $\mathcal{X}$ labelled $+1$ are the positive examples of $c$, whereas elements labelled $-1$ are the negative examples. Let $|\mathcal{X}| = N$ and let $\mathbf{w}$ be a probability distribution over $\mathcal{X}$. Let moreover $2^{\mathcal{X}}$ be the power set of $\mathcal{X}$. The learner $\mathcal{A}$, using a learning set $\mathbf{L}$, acquires an intensional description $\lambda$ of $c$, expressed in a given *hypothesis language*. Each description $\lambda$ classifies the elements of $\mathcal{X}$ partly as positive and partly as negative. Let $\varphi$ be the *extension* of $\lambda$, i.e., the set of examples that it classifies as positive. Obviously $\varphi \in 2^{\mathcal{X}}$. Let us now consider a set $\mathcal{L}$ of training sets. We can define $\mathcal{A}$ as a function from training sets to (extensional) hypotheses:

$$\mathcal{A} : \mathcal{L} \rightarrow 2^{\mathcal{X}} \tag{1}$$

Expression (1) represents the composition of two functions: the first one maps training sets into formulas of the hypothesis language, whereas the second one maps formulas to their extensions. As several descriptions in the hypothesis language may share the same extension, function (1) induces over the elements of $2^{\mathcal{X}}$ a probability distribution $\mathbf{q}$, which depends upon the elements in $\mathcal{L}$, and upon their probability distribution. We notice that, even though the description language may be infinite, the codomain of $\mathcal{A}$ in (1) is always finite, because $\mid 2^{\mathcal{X}} \mid = 2^N$. Finally, given $\mathcal{L}$, it is possible that not all the elements of $2^{\mathcal{X}}$ be reachable, i.e., it is possible that $\mathcal{A}$ can only generate a proper subset $\Phi$ of $2^{\mathcal{X}}$. Then, we may think of $\mathbf{q}$ as a probability distribution over $\Phi$, being the probabilities associated to the elements of $2^{\mathcal{X}} - \Phi$ equal to zero.

As we are neither interested in the specific working of $\mathcal{A}$ nor in the description language, let us abstract away, for the moment, the details of the learning process, and consider as given by an oracle the set $\Phi$ of hypotheses and the associated probability distribution $\mathbf{q}$. Let $|\Phi| = R$. Then, learning can be simulated by extracting with replacement from $\Phi$ hypotheses according to $\mathbf{q}$. In the following we will also consider as given the whole set $\mathcal{X}$ of examples, together with

[1] The finiteness of $\mathcal{X}$ is not essential; it is only assumed for the sake of simplicity.

**Table 1.** Matrix $\mathcal{M}$, describing the learning problem setting

| | $\varphi_1$ | .... | $\varphi_j$ | .... | $\varphi_R$ | |
|---|---|---|---|---|---|---|
| $x_1$ | $p_1(x_1)$ | | $p_j(x_1)$ | | $p_R(x_1)$ | $p(x_1)$ |
| .... | | | | | | |
| $x_k$ | $p_1(x_k)$ | .... | $p_j(x_k)$ | .... | $p_R(x_k)$ | $p(x_k)$ |
| .... | | | | | | |
| $x_N$ | $p_1(x_N)$ | .... | $p_j(x_N)$ | .... | $p_R(x_N)$ | $p(x_N)$ |
| | $r_1$ | | $r_j$ | | $r_R$ | $r$ |

its distribution $\mathbf{w}$. This is clearly an ideal case in which complete information is available.

Let us represent the described situation by means of the matrix $\mathcal{M}$ reported in Table 1. Given a hypothesis $\varphi_j$ and an example $x_k$, the classification $\varphi_j(x_k) \in \mathcal{Y} = \{+1, -1\}$ assigned by $\varphi_j$ to $x_k$ may be either correct or incorrect. Let $p_j(x_k) \in \{1, 0\}$[2] be the probability that hypothesis $\varphi_j$ correctly classifies example $x_k$, i.e., that $\varphi_j(x_k) = c(x_k)$. Let $p(x_k)$ be the average of such probabilities for $x_k$, and $r_j$ the accuracy of hypothesis $\varphi_j$ over the whole $\mathcal{X}$ ($r_j$ is the true error rate of $\varphi_j$). We have:

$$p(x_k) = \sum_{j=1}^{R} q_j p_j\,(x_k) \qquad r_j = \sum_{k=1}^{N} w_k p_j\,(x_k) \tag{2}$$

We can compute the average accuracy for the whole table:

$$r = \sum_{k=1}^{N} w_k p(x_k) = \sum_{j=1}^{R} q_j r_j = \sum_{j=1}^{R}\sum_{k=1}^{N} q_j w_k p_j(x_k) \tag{3}$$

In order to draw a parallel between Monte Carlo algorithms and ensemble learning, we note that the set $\Pi$ coincides with $\mathcal{X}$; in fact, each example $x_k \in \mathcal{X}$ is a problem to solve. Moreover, $\mathcal{Y}$ = {+1, -1} is the set of answers. In order to facilitate the analysis, let us permute the rows and the columns of $\mathcal{M}$ in such a way that the $p(x_k)$ values decrease from top to bottom and the $r_j$ values decrease from left to right. This simply amounts to rename both the examples and the hypotheses. Let us define the following Monte Carlo algorithm $\mathbf{MC}(x_k|\Phi, \mathbf{q})$, applied to a problem $x_k$:

[2] This assumption implies that the Bayes error is zero.

**MC**$(x_k|\Phi, \mathtt{q})$
  `Extract` $\varphi_j$ `from` $\Phi$ `according to q`
  `Classify` $x_k$ `with` $\varphi_j$
  `Return` $\varphi_j(x_k)$

As $p(x_k)$ is the probability of extracting a hypothesis $\varphi_j$ that correctly classifies $x_k$, **MC** is $p(x_k)$-correct. Let $\gamma_k = p(x_k) - 1/2$ be the advantage of **MC**. If we make $T$ independent calls to **MC** and take the majority answer, we can let the advantage $\gamma_k$ become as close to 1/2 as desired, provided that:

- **MC** is consistent
- the calls to **MC** are independent from each other,
- $p(x_k) > 1/2$.

The first condition is true as long as the Bayes error is zero (in this case $x_k$ has only one correct label). The second condition is more tricky: the calls to **MC** are independent by definition, but nothing assures us that the $\varphi_j$'s themselves are not related each other. This depends on the way $\mathcal{A}$ exploits the learning set to generate the hypotheses. We can expect that the more independent the $\varphi_j$'s are, the better the repeated application of **MC** will provide results close to the theoretical predictions.

Let now $\mathcal{X}_a = \{x_k | p(x_k) > 1/2\}$ be the subset of $\mathcal{X}$ containing the *amplifiable* examples, i.e., those that will be correctly classified in the long run.

If we define by $\pi_t(x_k)$ the probability that $x_k$ is correctly classified by $T$ calls to **MC**, we obtain:

$$\pi_T(x_k) = \sum_{t=\lfloor \frac{T}{2} \rfloor + 1}^{T} \binom{T}{t} p(x_k)^t (1 - p(x_k))^{T-t} \tag{4}$$

Then

$$\forall x_k \in \mathcal{X}_a : \lim_{T \to \infty} \pi_T(x_k) = 1 \tag{5}$$

In Figures 1,2, and 3, some graphs of $\pi_T(x_k)$ versus $T$ are reported. From these figures one can see that $p(x_k)$ has a strong effect on the number of **MC** calls needed to let $\pi_T(x_k)$ increase.

From (5) it is clear that the probability measure

$$\|\mathcal{X}_a\| = \sum_{x_k \in \mathcal{X}_a} w_k =_{\text{def}} \rho_\infty$$

is the upper bound of the asymptotic accuracy. This accuracy will be 1 when $\mathcal{X}_a = \mathcal{X}$.

By using the notions introduced so far, we can define the following Monte Carlo ensemble learning algorithm, **AmpMC**:

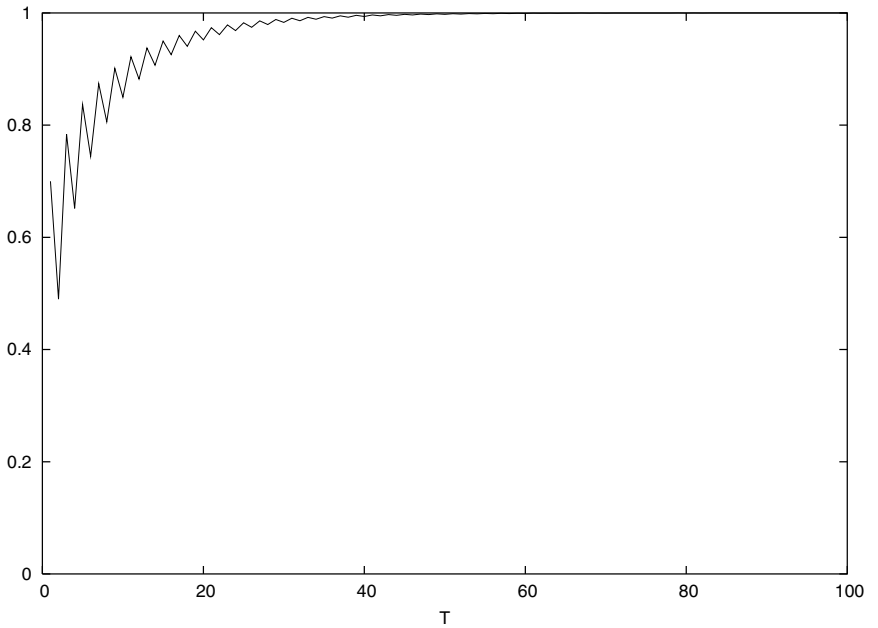



**Fig. 1.** Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.7$

**AmpMC**$(\Phi, \mathcal{X}, T)$
  `Extract with replacement` $x_k$ `from` $\mathcal{X}$ `according to` $\mathbf{w}$
  $H_T = 0$
  **for** $t = 1$, $T$ **do**
    $h_t = \mathbf{MC}(x_k|\Phi, \mathbf{q}) = \varphi_t(x_k)$
    $H_T = H_T + h_t$
  **end**
  **if** $H_T > 0$ **then** $y(x_k) = 1$ **else** $y(x_k) = -1$

It is important to point out that algorithm **MC** runs $T$ times over each example $x_k$, and the results of the runs on previous examples are forgotten. In a sense, **AmpMC** works "by rows" on $\mathcal{M}$, and the classification accuracy of the base learner is not taken into consideration explicitly, but only indirectly through (3). On the contrary, bagging works, so to say, "by columns", by extracting $T$ columns from $\Phi$ and using them to obtain a majority vote on each $x_k$. Considering the global average accuracy $r$, or even the single $r_j$'s, it is not immediate to say how many of the $x_k$'s will be correctly classified by the majority voting procedure.

By establishing a link between bagging's and **AmpMC**'s behaviors, we want to see whether this latter can explain (or better, predicts) the former. In this paper we limit ourselves to the explanation aspect. If we extract $T$ columns $\varphi_t$ $(1 \leq t \leq T)$ from $\Phi$, independently and without replacement, we see that, for

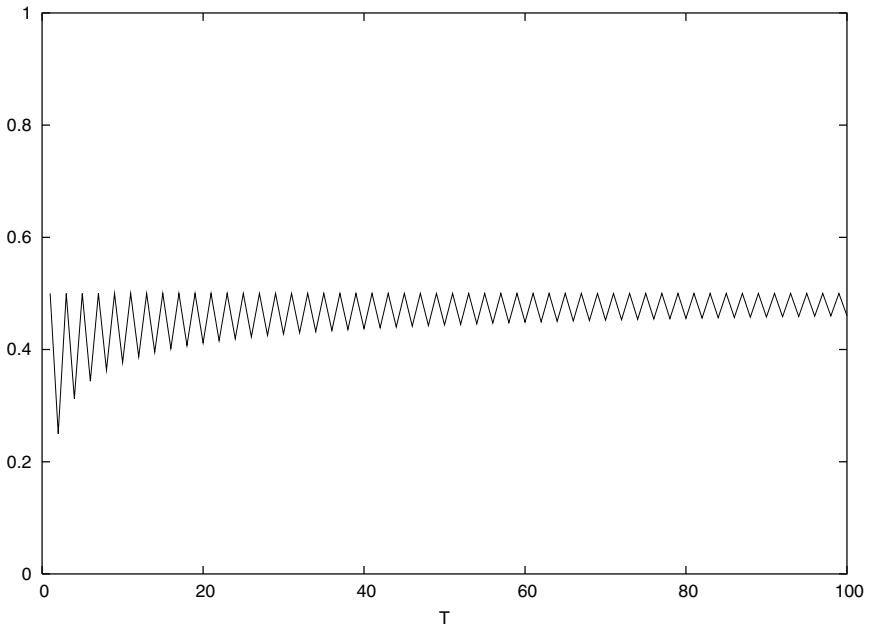



**Fig. 2.** Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.5$

each $x_k \in \mathcal{X}$ the conditions 1) and 2) required by the Monte Carlo theory are still verified, but condition 3) is not guaranteed, not even in the case that all $r_t$ are greater than 1/2. In fact, as the $r_t$'s and the $p(x_k)$'s are only loosely coupled by (3), it is possible that $\mathcal{X}_a$ be a proper subset of $\mathcal{X}$ and, then, not all examples can be amplified. It is easy to prove that $\rho_\infty$ can reach at most the value $2r$ [8].

Thus, even with single hypotheses with accuracy less then 1/2 it is possible to obtain higher accuracies by Monte Carlo amplification. However, in order to reach $\rho_\infty = 1$, the value of $r$ must be at least 1/2. Even in this case, some of the component hypotheses are allowed to have an accuracy less than 1/2, without hindering amplification, provided that $\mathcal{X}_a = \mathcal{X}$ and that $r$ does not go below 1/2.

## 4 Predicting Bagging's Behavior

In previous sections we claimed that Bagging can be viewed as the application of the Monte Carlo algorithms theory to Machine Learning. Here we use the Monte Carlo theory to estimate the empirical error of the bagged classifier at time $t$, which is important in two different ways. First, experiments show that the estimate is indeed *very* accurate, offering, thus, an empirical support to the claim we made. Second, it has a significant theoretical meaning, since it makes clear what process is going on as learning proceeds. The estimate assumes

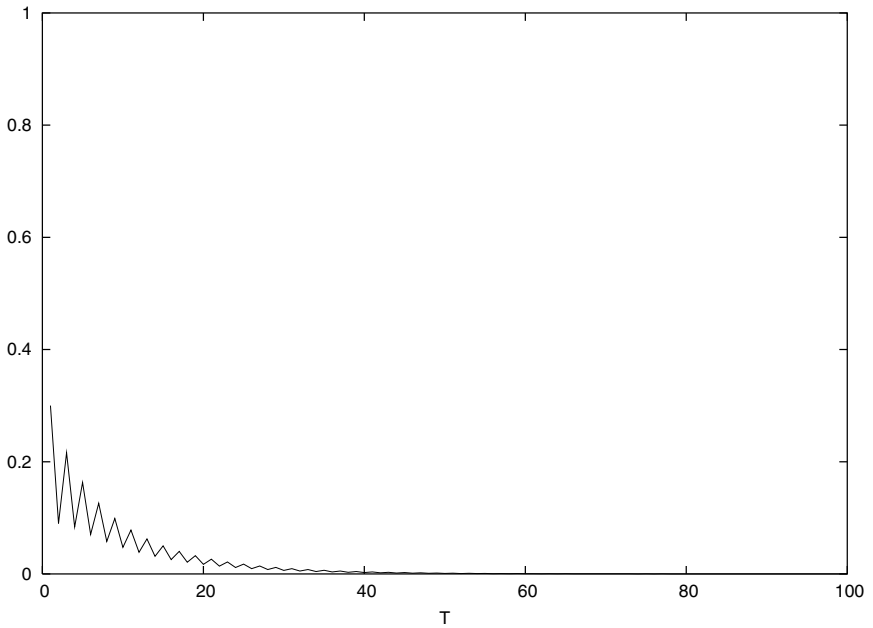



**Fig. 3.** Graph of $\pi_T(x_k)$ vs. T for $p(x_k) = 0.3$

that $\mathbf{p} = (p(\boldsymbol{x}_1), p(\boldsymbol{x}_2), \ldots, p(\boldsymbol{x}_m))$ is known. Actually, in practice $\mathbf{p}$ is, in turn, estimated empirically.

Let us denote with $\mathcal{D}(p)$ the probability distribution of $p(\boldsymbol{x})$. Monte Carlo theory explains Bagging's behavior in terms of $\mathcal{D}(p)$. In particular, when the weak learning algorithm has been chosen, the distribution of $p(\boldsymbol{x})$ can be considered as a "property" of the examples themselves and Bagging as a way to "discover" such a property. As a consequence, the error committed by the combined classifier when the number of combined hypotheses is $T$ can be decomposed into three parts (let us denote with $\hat{p}(\boldsymbol{x})$ the observed $p(\boldsymbol{x})$ after $T$ iterations of the weak learning algorithm):

- the final accuracy $\rho_\infty$ of the Monte Carlo process, i.e., the accuracy of the combined classifier when $T \to \infty$. In terms of $\mathcal{D}(p)$, this term can be written as:
$$\rho_\infty = \int_{p>0.5} \mathcal{D}(p)\, dp$$
- A correction $\rho_T^-$ to the final accuracy, given by the fraction of examples for which $p(\boldsymbol{x}) \leq 0.5$, but which, with $T$ hypotheses, are likely to be overestimated so that the apparent $\hat{p}(\boldsymbol{x})$ is greater than 0.5.
- A correction $\rho_T^+$ to the final accuracy, given by the fraction of examples for which $p(\boldsymbol{x}) > 0.5$, but which, with $T$ hypotheses, are likely to be underestimated so that the apparent $\hat{p}(\boldsymbol{x})$ is less than 0.5.

By summarizing, the expected accuracy of the Bagging algorithm with $T$ hypotheses is:

$$\rho_T = \rho_\infty - \rho_T^+ + \rho_T^- \tag{6}$$

Since the value $\rho_\infty$ is fixed in advance and does not change, as more hypotheses are combined, the drop in the error of the bagged hypothesis is explained because the other two terms tend to zero. Actually, it is not so hard to reduce $\rho_T^+$ and $\rho_T^-$ to something that is simple to compute, but we omit the details here for the sake of simplicity.

The above argumentation builds upon the assumption that nothing more than a Monte Carlo process is going on during the learning process. A preliminary experimentation has been performed in order to test the above conjecture.

The experiment consists of two parts. In the first one, we acquire $T = 1000$ hypotheses using the Bagging algorithm. The training error of the bagged hypothesis is measured during the learning process for each $1 \leq T \leq 1000$. At the end of the process, for $T = 1000$, we use the collected hypotheses to estimate the distribution $\mathcal{D}(p)$ of $p(\boldsymbol{x})$.

In the second phase, we perform a straightforward amplification of a Monte Carlo algorithm for which the probability to solve a particular problem follows $\mathcal{D}(p)$, i.e., the distribution that we estimated in the first part of the experiment. In particular, we built a Monte Carlo procedure $\mathbf{MC}(p)$ which simply extracts at random a number in [0..1] and pretends to be correct whenever that number is less than $p$. The amplified version of the algorithm performs the experiment $T$ times and is correct whenever more than $T/2$ experiments were successful. As we did for Bagging, we measure the accuracy of the algorithm for $1 \leq T \leq 1000$.

Figures 4 and 5 show the errors of the two algorithms plotted against increasing $T$ values. The two experiments were performed using the "ionosphere" dataset from Irvine's repository and a two dimensional, noise free, dataset that we call "Asia". In both cases the weak learning algorithm is a simple CART-like algorithm[3].

In [8] and [9] the correspondence between Bagging and Monte Carlo theory has been exploited also to estimate Bagging's generalization error, and a much larger experimentation is presented. Here, we want to emphasize the meaning of the presented experiments. In particular, it is important to note that while the first experiment has access to all learning parameters (such as the weak learner, the hypotheses it generates, the training set, etc.), the second one can access only few of them, namely the training set size, and the distribution $\mathcal{D}(p)$. Moreover, we note that $\mathcal{D}(p)$ is related to the final bagged hypothesis only. In fact, it reports, for each $p \in [0 \ldots 1]$, the fraction of examples for which the observed frequencies at $T = 1000$ is $p$. While such information allows one to immediately evaluate the accuracy of the final bagged classifier, its error when $T \neq 1000$ cannot be evaluated (as far as we know) as a function of $\mathcal{D}(p)$.

[3] In the implementation used, the algorithm stops to grow the tree as soon as eighty percent of the examples are covered.

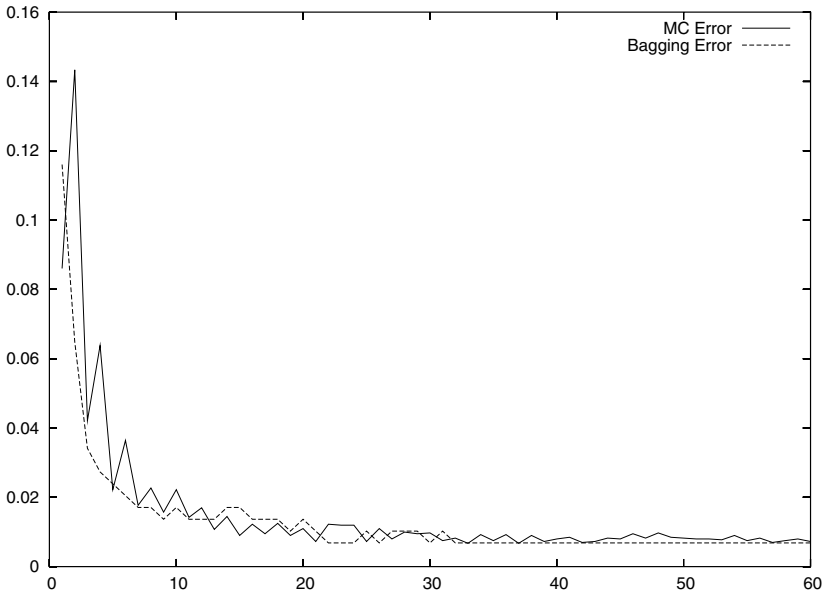



**Fig. 4.** Plot of the errors of a simulated Monte Carlo algorithm and the errors of Bagging on the "Ionosphere" dataset

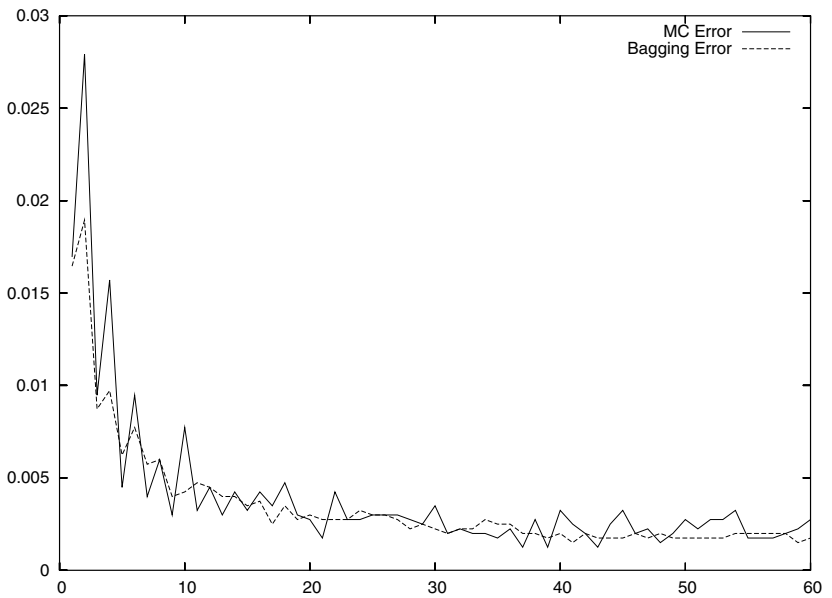



**Fig. 5.** Plot of the errors of a simulated Monte Carlo algorithm and the errors of Bagging on an artificial dataset

## 5 Monte Carlo Interpretation of Examples' Margin

In Ensemble Learning the notion of margin has been introduced for the first time by Schapire et al. [16] as an explanation of the effectiveness of their AdaBoost algorithm. They define the weight assigned to a label as the (weighted) fraction of hypotheses that output that label.

**Definition 5.** (Schapire et al. [16]) *For any particular example $x_k$, the classification* margin $\mu(x_k)$ *for $x_k$ is the difference between the weight assigned to the correct label and the maximal weight assigned to any single incorrect label.*

**Theorem 2.** *In the case of binary classification, the following relation holds between the margin and the Monte Carlo $p(\boldsymbol{x})$ value:*

$$\mu(\boldsymbol{x}) = p(\boldsymbol{x}) - (1 - p(\boldsymbol{x})) = 2p(\boldsymbol{x}) - 1$$

The proof of Theorem 2 derives immediately from the definition of margin and $p(\boldsymbol{x})$. The theorem directly explains why increasing the margin improves the performances of boosting: in fact, increasing the minimum margin lets the cardinality of the set $\mathcal{X}_a$ increase, augmenting the number of examples for which the Monte Carlo conditions for amplifiability hold.

Interestingly, formula (6) implies that Bagging does not increase the margin at all. More precisely, it "uncovers" the margin of each example. This is rather different from AdaBoost approach, which directly maximizes the minimal margin over the examples.

## 6 Conclusion

In this paper we have proposed to analyze ensemble learning (in particular Bagging) from the perspective of Monte Carlo algorithms theory. In particular, after introducing basic Monte Carlo theory notions, we showed that these notions, interpreted from a Machine Learning perspective, directly lead to the Bagging algorithm. The paper then focused on the implications of the link. In particular, it emphasized that the real important parameter is the probability with which each example is correctly classified, and explained why reducing the margin increases the algorithm performances. Finally, the paper offered a preliminary experimentation supporting the claim that Bagging is indeed nothing more than a Monte Carlo algorithm.

## References

[1] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999. 190

[2] Gilles Brassard and Paul Bratley. *Algorithmics: theory and practice.* Prentice-Hall, Inc., 1988. 190, 191

[3] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 189, 190

[4] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic regression, adaboost and bregman distances. In *Computational Learing Theory*, pages 158–169, 2000. 189

[5] Thomas G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857:1–15, 2000. 189, 190

[6] Harris Drucker and Corinna Cortes. Boosting decision trees. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 479–485. The MIT Press, 1996. 189

[7] B. Efron and R. Tibshirani. *An introduction to the Bootstrap*. Chapman and Hall, 1993. 190

[8] Roberto Esposito. *Analyzing Ensemble Learning in the Framework of Monte Carlo Theory*. PhD thesis, Dipartimento di Informatica, Università di Torino, C.so Svizzera 185, 10149 Torino, Italy, 2003. 195, 197

[9] Roberto Esposito and Lorenza Saitta. Monte Carlo Theory as an Explanation of Bagging and Boosting. In *Proceeding of the Eighteenth International Joint Conference on Artificial Intelligence (to appear)*, 2003. 197

[10] Y. Freund and R. Schapire. Game theory, on-line prediction and boosting. In *Proceedings, 9th Annual Conference on Computational Learning Theory*, pages 325–332, 1996. 189

[11] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th International Conference on Machine Learning*, pages 148–146. Morgan Kaufmann, 1996. 189

[12] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1(55):119–139, 1997. 189

[13] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. Technical report, Department of Statistics, Sequaoia Hall, Stanford University, 1998. 189

[14] J. R. Quinlan. Bagging, boosting and c4.5. In *Proc. of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 725–730. AAAI Press / MIT Press, 1996. 190

[15] Gunnar Rätsch, Manfred Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus-Robert Müller. Barrier boosting. In *Thirteenth Annual Conference on Computational Learning Theory*, 2000. 189

[16] R. Scha-pire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistic*, 26(5):1651–1686, 1998. 189, 199

[17] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. 189

[18] Robert E. Schapire. Drifting games. In *Computational Learing Theory*, pages 114–124, 1999. 189

[19] David H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992. 189

# Pairwise Data Clustering Using Monotone Game Dynamics

Massimiliano Pavan and Marcello Pelillo

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Via Torino 155, 30172 Venezia Mestre, Italy
{mapavan,pelillo}@dsi.unive.it

**Abstract.** We propose a new framework for the pairwise data clustering problem which is based on a "standard" quadratic program formulation, i.e., a continuous quadratic optimization problem with simplex (or probability) constraints. We then introduce a wide family of dynamic equations from evolutionary game theory, known as *payoff-monotonic* dynamics, and study their dynamical properties. These properties make any member of this family a potential heuristic for solving standard quadratic programs and, in particular, our data clustering problem. We demonstrate the potential of our framework for the problem of the unsupervised learning of shape categories from an image database. Experiments with two different similarity matrices (and databases) reported in the computer vision literature have been conducted, and the results confirm the effectiveness of our approach.

## 1 Introduction

The unsupervised partitioning of data, or *clustering*, is a pervasive problem in machine learning and pattern recognition [2, 8]. Conventional, *central* clustering methods (such as, e.g., $k$-means and square-error algorithms), which map the data items to some real normed vector space (called *feature space*), have been proven to play an important role from both a theoretical and a practical standpoint [8]. However, feature-based approaches have the disadvantage of requiring feature selection, which is often an elusive task. Moreover, they suffer from the inadequacy of the metric utilized in parameter space to appropriately represent (dis)similarities between data items.

Formulating the clustering problem in a *pairwise* setting is a common way to overcome these problems [7]. Pairwise data clustering methods map (possibly implicitly) the data items to be clustered to the nodes of a weighted graph, with edge weights representing similarity or dissimilarity relations. Although this representation lacks geometrical notions such as scatter and centroids, it is attractive as no feature selection is required and pairwise relations are also allowed to violate metric properties such as the triangular inequality, a situation that cannot be modeled when data are embedded in a normed space. Thanks to this flexibility, pairwise clustering methods are gaining increasing popularity in

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 201–212, 2003.
© Springer-Verlag Berlin Heidelberg 2003

domains such as computer vision and pattern recognition (see, e.g., [20, 7, 17, 10, 19]).

In this paper, we propose a novel approach to pairwise data clustering problems. We formulate the problem in terms of a "standard" quadratic program, namely a continuous quadratic optimization problem with simplex (or probability) constraints, and show that the solutions of this program are in tight correspondence with clusters of data items. Indeed, in a previous work we have also provided a combinatorial interpretation of these solutions in terms of a newly introduced graph-theoretic notion [13]. The proposed objective function, which quantifies the degree of cohesiveness of a cluster, has also been recently used by other authors, but using different constraints [19, 17], and our program is a generalization of the so-called Motzkin-Straus problem from graph theory [12].

Our formulation of the clustering problem allows us to tackle it by means of dynamic equations from evolutionary game theory, a discipline pioneered by J. Maynard Smith [11] that aims to model the evolution of animal behavior using the principles and tools of noncooperative game theory [6]. Evolutionary game models are also gaining increasing popularity in economics since they allow one to elegantly get rid of the much-debated assumptions of traditional game theory concerning the full rationality and complete knowledge of players [22, 18, 3]. In particular, we focus on a wide family of evolutionary game dynamics characterized by having the growth rates of strategies ordered by their expected payoffs, so that strategies associated to higher payoffs grow faster. These *payoff-monotonic* dynamics, which have already proven to be effective in graph matching and related problems [15, 16], are attractive as can be coded in a few lines of any high-level programming language, can easily be implemented in a parallel network of locally interacting units, and offer the advantage of biological plausibility [23]. We show that, when the payoff matrix is symmetric, these models possess a quadratic Liapunov function which is strictly increasing along any nonconstant trajectory, and their asymptotically stable stationary points are in one-to-one correspondence with (strict) local solutions of standard quadratic programs. These properties make therefore *any* member of the payoff-monotonic family a potential heuristic for our pairwise data clustering problem.

We demonstrate the potential of our framework for the problem of the unsupervised learning of shape categories from an image database. Experiments with two similarity matrices (and databases) reported in the computer vision literature have been conducted, and the results confirm the power of the approach in correctly discovering data categories, despite its simplicity.

## 2 Clustering as Standard Quadratic Optimization

We represent the data to be clustered as an undirected edge-weighted (or similarity) graph with no self-loops $G = (V, E, w)$, where $V = \{1, \ldots, n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set, and $w : E \to \mathbb{R}_+^*$ is the (positive) weight function. Vertices in $G$ correspond to data points, edges represent neighborhood relationships, and edge-weights reflect similarity between pairs of linked vertices.

As customary, we represent the graph $G$ with the corresponding weighted adjacency (or similarity) matrix, which is the $n \times n$ symmetric matrix $A_G = (a_{ij})$ defined as:

$$a_{ij} = \begin{cases} w(i,j) \ , & \text{if } (i,j) \in E \\ 0 \ , & \text{otherwise.} \end{cases}$$

A common way to represent a cluster of nodes (see, e.g., [19, 17, 20]) is to associate a real $n$-dimensional vector to it, where its components express the participation of nodes in the cluster: if a component has a small value, then the corresponding node is weakly associated with the cluster, whereas if it has a large value, the node is strongly associated with the cluster. Components corresponding to nodes not participating in the cluster are zero.

Intuitively, a good cluster is one where elements that are strongly associated with it also have large values connecting one another in the similarity matrix. Hence, a natural way of defining the cohesiveness of a cluster is given by the following function (where a prime denotes transposition):

$$f_G(\mathbf{x}) = \mathbf{x}' A_G \mathbf{x} \tag{1}$$

and this allows us to formulate the (pairwise) clustering problem as the problem of finding a vector $\mathbf{x}$ that maximizes $f_G$. However, note that the objective function is useless without some normalization of the components of $\mathbf{x}$, and thus we impose to it simplex (or probability) constraints. This yields the following standard quadratic program:

$$\begin{array}{ll} \text{maximize} & f_G(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \Delta \end{array} \tag{2}$$

where

$$\Delta = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0} \text{ and } \mathbf{e}'\mathbf{x} = 1\}$$

is the standard simplex of the $n$-dimensional Euclidean space $\mathbb{R}^n$ ($\mathbf{e}$ is a vector of appropriate length consisting of unit entries). We note that program (2) includes as a special case the one introduced by Motzkin and Straus [12], which is intimately related to the maximum clique problem, a classical $NP$-hard problem from graph theory.

Thus, in our formulation, a maximally cohesive cluster can be found by determining a solution of program (2). Specifically, a point $\mathbf{x}^* \in \Delta$ is said to be a *local* solution of program (2) if there exists an $\varepsilon > 0$ such that $f_G(\mathbf{x}^*) \geq f_G(\mathbf{x})$ for all $\mathbf{x} \in \Delta$ whose distance from $\mathbf{x}^*$ is less than $\varepsilon$, and if $f_G(\mathbf{x}^*) = f_G(\mathbf{x})$ implies $\mathbf{x}^* = \mathbf{x}$, then $\mathbf{x}^*$ is said to be a *strict* local solution.

We have recently shown that (strict) local solutions of program (2) can be given an interesting combinatorial interpretation in terms of a newly introduced graph-theoretic concept, that of a *dominant set* [13]. As thoroughly discussed in [13], dominant sets are a natural extension of the notion of a (strictly) maximal clique to edge-weighted graphs. In fact, the two concepts turn out to be equivalent when the similarity matrix consists of zeros and ones or, put another way, when the underlying graph is unweighted. Note that maximal cliques are

a classic formalization of the notion of a cluster [8], and this provides further evidence as to the validity of our formulation of the clustering problem.

A similar (though substantially different) formulation to pairwise clustering has recently been proposed by Sarkar and Boyer [19] (see also [17]), who suggested maximizing $\mathbf{x}' A_G \mathbf{x}$ subject to $\mathbf{x}'\mathbf{x} = 1$, leading to a continuous quadratic optimization problem over the sphere instead of the simplex. Interestingly, this is equivalent to determining the eigenvalues (and corresponding eigenvectors) of the similarity matrix $A_G$, a problem for which standard algorithms exist. However, this "spectral" approach has a serious drawback, in that in general the components of an eigenvector are not necessarily nonnegative. Eigenvectors with negative components are meaningless in the above interpretation since the membership of a node in a cluster cannot be negatively valued. This lead Sarkar and Boyer, somewhat arbitrarily, to consider only same-sign eigenvectors, thereby ignoring the others. As an aside, we also note that no combinatorial interpretation is offered for their "eigenclusters."

Once the clustering problem is formulated as an optimization problem, we can use any optimization technique to solve it. In the next sections, we shall see that payoff-monotonic dynamics from evolutionary game theory lend themselves well to this task.

## 3 Payoff-Monotonic Game Dynamics

Evolutionary game theory considers an idealized scenario whereby in a large population pairs of individuals are repeatedly drawn at random to play a symmetric two-player game. In contrast to traditional game theoretic models, players are not supposed to behave rationally or to have complete knowledge of the details of the game. They act instead according to a pre-programmed behavior pattern, or pure strategy, and it is supposed that some evolutionary selection process operates over time on the distribution of behaviors. We refer the reader to [6, 22] for excellent introductions to this rapidly expanding field.

Let $J = \{1, \cdots, n\}$ be the set of available pure strategies and, for all $i \in J$, let $x_i(t)$ be the proportion of population members playing strategy $i$, at time $t$. The state of the population at a given instant is the vector $\mathbf{x} = (x_1, \cdots, x_n)'$. Clearly, population states are constrained to lie in the standard simplex $\Delta$ of $\mathbb{R}^n$ as defined in the previous section. The *support* of a population state $\mathbf{x} \in \Delta$, denoted by $\sigma(\mathbf{x})$, is defined as the set of indices corresponding to its positive components, which correspond to non-extinct strategies:

$$\sigma(\mathbf{x}) = \{i \in J \ : \ x_i > 0\} \ .$$

Given a subset of strategies $S \subseteq J$, the set of states where all strategies outside $S$ are extinct, which corresponds to a face of $\Delta$, is defined as:

$$\Delta_S = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) \subseteq S\}$$

and its (relative) interior is:

$$\mathrm{int}(\Delta_S) = \{\mathbf{x} \in \Delta \ : \ \sigma(\mathbf{x}) = S\} \ .$$

Clearly, $\Delta_J = \Delta$ and, accordingly, we shall write $\mathrm{int}(\Delta)$ instead of $\mathrm{int}(\Delta_J)$.

Let $A = (a_{ij})$ be the $n \times n$ payoff or utility matrix. Specifically, for each pair of strategies $i, j \in J$, $a_{ij}$ represents the payoff of an individual playing strategy $i$ against an opponent playing strategy $j$. In biological contexts payoffs are typically measured in terms of Darwinian fitness or reproductive success (i.e., the player's expected number of offspring), whereas in economic applications they usually represent firms' profits or consumers' utilities. If the population is in state $\mathbf{x}$, the expected payoff earnt by an $i$-strategist is:

$$\pi_i(\mathbf{x}) = \sum_{j=1}^{n} a_{ij} x_j = (A\mathbf{x})_i \tag{3}$$

while the mean payoff over the entire population is

$$\pi(\mathbf{x}) = \sum_{i=1}^{n} x_i \pi_i(\mathbf{x}) = \mathbf{x}' A \ \mathbf{x} \ . \tag{4}$$

In evolutionary game theory the assumption is made that the game is played over and over, generation after generation, and that the action of natural selection will result in the evolution of the fittest strategies. If successive generations blend into each other, the evolution of behavioral phenotypes can be described by a set of ordinary differential equations. A general class of evolution equations is given by:

$$\dot{x}_i = x_i g_i(\mathbf{x}) \tag{5}$$

where a dot signifies derivative with respect to time, and $g = (g_1, \ldots, g_n)$ is a function with open domain containing $\Delta$. Here, the function $g_i$ $(i \in j)$ specifies the rate at which pure strategy $i$ replicates when the population is in state $\mathbf{x}$. It is usually required that the growth function $g$ is *regular* [22], which means that it is $C^1$ and that $g(\mathbf{x})$ is always orthogonal to $\mathbf{x}$, i.e., $g(\mathbf{x})'\mathbf{x} = 0$. The former condition guarantees us that the system of differential equations (5) has a unique solution through any initial population state. The condition $g(\mathbf{x})'\mathbf{x} = 0$, instead, ensures that the simplex $\Delta$ is invariant under (5), namely any trajectory starting in $\Delta$ will remain in $\Delta$.

A point $\mathbf{x}$ is said to be a *stationary* (or equilibrium) point of our dynamical system, if $\dot{x}_i = 0$, for all $i = 1 \ldots n$. A stationary point $\mathbf{x}$ is said to be *Liapunov stable* (or, more simply *stable*) if for any neighborhood $U$ of $\mathbf{x}$ there exists a neighborhood $W$ of $\mathbf{x}$ such that any trajectory in $W$ remains also in $U$ (formally, $\mathbf{x}(0) \in W$ implies $\mathbf{x}(t) \in U$ for all $t \geq 0$). It is said to be *asymptotically stable* if, in addition, such trajectories converge to $\mathbf{x}$.

Payoff-monotonic game dynamics represent a wide class of regular selection dynamics for which useful properties hold. Intuitively, for a payoff-monotonic dynamics the strategies associated to higher payoffs will increase at higher rate. Formally, a regular selection dynamics (5) is said to be payoff-monotonic if:

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \ \Leftrightarrow \ \pi_i(\mathbf{x}) > \pi_j(\mathbf{x}) \tag{6}$$

for all $\mathbf{x} \in \Delta$.

Although this class contains many different dynamics, it turns out that they share many common properties. To begin with, they all have the same set of stationary points.

**Proposition 1.** *A point* $\mathbf{x} \in \Delta$ *is stationary under any payoff-monotonic dynamics if and only if* $\pi_i(\mathbf{x}) = \pi(\mathbf{x})$ *for all* $i \in \sigma(\mathbf{x})$.

*Proof.* See [22, Proposition 4.7]. □

A well-known subclass of payoff-monotonic game dynamics is given by:

$$\dot{x}_i = x_i \left( \phi(\pi_i(\mathbf{x})) - \sum_{j=1}^{n} x_j \phi(\pi_j(\mathbf{x})) \right) \tag{7}$$

where $\phi(u)$ is an increasing function of $u$. These models arise in modeling the evolution of behavior by way of imitation processes, where players are occasionally given the opportunity to change their own strategies [5, 22].

When $\phi$ is the identity function, i.e., $\phi(u) = u$, we obtain the standard first-order *replicator equations* [22, 6]:

$$\dot{x}_i = x_i \left( \pi_i(\mathbf{x}) - \sum_{j=1}^{n} x_j \pi_j(\mathbf{x}) \right) \tag{8}$$

whose basic idea is that the average rate of increase $\dot{x}_i/x_i$ equals the difference between the average fitness of strategy $i$ and the mean fitness over the entire population.

Another popular model arises when $\phi(u) = e^{\kappa u}$ which yields:

$$\dot{x}_i = x_i \left( e^{\kappa \pi_i(\mathbf{x})} - \sum_{j=1}^{n} x_j e^{\kappa \pi_j(\mathbf{x})} \right) \tag{9}$$

where $\kappa$ is a positive constant. As $\kappa$ tends to 0, the orbits of this dynamics approach those of the standard, first-order replicator model (8), slowed down by the factor $\kappa$; moreover, for large values of $\kappa$ the model approximates the so-called "best-reply" dynamics [5, 6].

## 4 Payoff-Monotonic Dynamics and Quadratic Programming

In this section we shall explore the connections between payoff-monotonic dynamics and quadratic optimization problems. Consider the following standard quadratic program:

$$\begin{array}{ll} \text{maximize} & \pi(\mathbf{x}) = \mathbf{x}' A\, \mathbf{x} \\ \text{subject to} & \mathbf{x} \in \Delta \end{array} \tag{10}$$

where $A$ is an arbitrary $n \times n$ symmetric matrix. In evolutionary game theory a symmetric payoff matrix arises in the context of doubly symmetric (or partnership) games, where the interests of the two players coincide [6, 22].

A point $\mathbf{x} \in \Delta$ satisfies the Karush-Kuhn-Tucker (KKT) conditions for problem (10), i.e. the first-order necessary conditions for local optimality, if there exist $n+1$ real constants $\mu_1, \ldots, \mu_n$ and $\lambda$, with $\mu_i \geq 0$ for all $i = 1 \ldots n$, such that:

$$(A\mathbf{x})_i - \lambda + \mu_i = 0$$

for all $i = 1 \ldots n$, and

$$\sum_{i=1}^{n} x_i \mu_i = 0 \ .$$

Note that, since both $x_i$ and $\mu_i$ are nonnegative for all $i = 1 \ldots n$, the latter condition is equivalent to saying that $i \in \sigma(\mathbf{x})$ implies $\mu_i = 0$. Hence, the KKT conditions can be rewritten as:

$$(A\mathbf{x})_i \begin{cases} = \lambda & \text{if } i \in \sigma(\mathbf{x}) \\ \leq \lambda & \text{if } i \notin \sigma(\mathbf{x}) \end{cases} \tag{11}$$

for some real constant $\lambda$. On the other hand, it is immediate to see that $\lambda = \mathbf{x}' A\, \mathbf{x}$. A point $\mathbf{x} \in \Delta$ satisfying (11) will be called a *KKT point* throughout.

The following easily proved result establishes a first connection between standard quadratic programs and payoff-monotonic dynamics.

**Proposition 2.** *If $\mathbf{x} \in \Delta$ is a KKT point for (10), then it is a stationary point of any payoff-monotonic dynamics. If $\mathbf{x} \in \mathrm{int}(\Delta)$ then the converse also holds.*

*Proof.* A straightforward consequence of Proposition 1 and the KKT conditions (11). □

In an unpublished paper, Hofbauer [5] showed that for symmetric payoff matrices the population mean payoff $\mathbf{x}' A \mathbf{x}$ is strictly increasing along the trajectories of any payoff-monotonic dynamics. This result generalizes the celebrated "fundamental theorem of natural selection" [6, 22]. Here, we provide a different proof adapting a technique from [3].

**Theorem 1.** *If the payoff matrix $A$ is symmetric, then $\pi(\mathbf{x}) = \mathbf{x}' A \mathbf{x}$ is strictly increasing along any nonconstant trajectory of any payoff-monotonic dynamics. In other words, $\dot{\pi}(\mathbf{x}(t)) \geq 0$ for all $t$, with equality if and only if $\mathbf{x} = \mathbf{x}(t)$ is a stationary point.*

*Proof.* For $\mathbf{x} \in \Delta$ define:

$$\sigma_+(\mathbf{x}) = \{i \in \sigma(\mathbf{x}) \ : \ g_i(\mathbf{x}) \geq 0\}$$

and

$$\sigma_-(\mathbf{x}) = \{i \in \sigma(\mathbf{x}) \ : \ g_i(\mathbf{x}) < 0\} \ .$$

Clearly, $\sigma_+(\mathbf{x}) \cup \sigma_-(\mathbf{x}) = \sigma(\mathbf{x})$. Moreover, let:

$$\overline{\pi}(\mathbf{x}) = \min\{\pi_i(\mathbf{x}) \ : \ i \in \sigma_+(\mathbf{x})\}$$

and

$$\underline{\pi}(\mathbf{x}) = \max\{\pi_i(\mathbf{x}) \ : \ i \in \sigma_-(\mathbf{x})\}$$

Because of payoff-motonicity, note that $\overline{\pi}(\mathbf{x}) \geq \underline{\pi}(\mathbf{x})$. Note also that $\dot{x}_i \geq 0$ if and only if $i \in \sigma_+(\mathbf{x})$.

Because of the symmetry of $A$, we have:

$$\begin{aligned}
\frac{\dot{\pi}(\mathbf{x})}{2} &= \sum_{i \in \sigma(\mathbf{x})} \dot{x}_i \pi_i(\mathbf{x}) \\
&= \sum_{i \in \sigma_+(\mathbf{x})} \dot{x}_i \pi_i(\mathbf{x}) + \sum_{i \in \sigma_-(\mathbf{x})} \dot{x}_i \pi_i(\mathbf{x}) \\
&\geq \overline{\pi}(\mathbf{x}) \sum_{i \in \sigma_+(\mathbf{x})} \dot{x}_i + \underline{\pi}(\mathbf{x}) \sum_{i \in \sigma_-(\mathbf{x})} \dot{x}_i \\
&= (\overline{\pi}(\mathbf{x}) - \underline{\pi}(\mathbf{x})) \sum_{i \in \sigma_+(\mathbf{x})} \dot{x}_i \\
&\geq 0
\end{aligned}$$

where the last equality follows from $\sum \dot{x}_i = 0$. Finally, note that $\dot{\pi}(\mathbf{x}) = 0$ if and only if $\pi_i(\mathbf{x})$ is constant for all $i \in \sigma(\mathbf{x})$ which, from Proposition 1, amounts to saying that $\mathbf{x}$ is stationary. □

Apart from the monotonicity result which provides a (strict) Liapunov function for payoff-monotonic dynamics, the previous theorem also rules out complicated attractors like cycles, invariant tori, or even strange attractors. It also allows us to establish a strong connection between the stability properties of these dynamics and the solutions of (10). To this end, we need an auxiliary result.

**Lemma 1.** *Let* $\mathbf{x}$ *be a strict local solution of (10), and put* $S = \sigma(\mathbf{x})$. *Then,* $\mathbf{x}$ *is the only stationary point of any payoff-monotonic dynamics in* $\mathrm{int}(\Delta_S)$. *Moreover,* $\mathbf{x}'A\ \mathbf{x} > \mathbf{y}'A\ \mathbf{y}$ *for all* $\mathbf{y} \in \Delta_S$.

*Proof.* Clearly, since $\mathbf{x}$ is a strict local solution of (10), it is a KKT point and hence is stationary under any payoff-monotonic dynamics by Proposition 2. Suppose by contradiction that $\mathbf{y} \in \mathrm{int}(\Delta_S) \setminus \{\mathbf{x}\}$ is stationary too. Then, it is easy to see that all points on the segment joining $\mathbf{x}$ and $\mathbf{y}$, which is contained in $\mathrm{int}(\Delta_S)$ because of its convexity, consists entirely of stationary points. Hence, by Theorem 1, $\dot{\pi} = 0$ on this segment which means that $\pi$ is constant, but this contradicts the hypothesis that $\mathbf{x}$ is a strict local solution of (10).

Moreover, for a sufficiently small $\varepsilon > 0$ we have

$$\begin{aligned}
\mathbf{x}'A\ \mathbf{x} &> [(1-\varepsilon)\mathbf{x} + \varepsilon\mathbf{y}]'A[(1-\varepsilon)\mathbf{x} + \varepsilon\mathbf{y}] \\
&= (1-\varepsilon)^2\mathbf{x}'A\mathbf{x} + 2\varepsilon(1-\varepsilon)\mathbf{y}'A\mathbf{x} + \varepsilon^2\mathbf{y}'A\ \mathbf{y}
\end{aligned}$$

but, since $\mathbf{x}$ is stationary and $\sigma(\mathbf{y}) \subseteq \sigma(\mathbf{x})$, $\mathbf{y}'A\ \mathbf{x} = \mathbf{x}'A\ \mathbf{x}$, from which we readily obtain $\mathbf{x}'A\ \mathbf{x} > \mathbf{y}'A\ \mathbf{y}$. □

**Theorem 2.** *A point $\mathbf{x} \in \Delta$ is a strict local solution of program (10) if and only if it is asymptotically stable under any payoff-monotonic dynamics.*

*Proof.* If $\mathbf{x}$ is asymptotically stable then there exists a neighborhood $U$ of $\mathbf{x}$ in $\Delta$ such that every trajectory starting in a point $\mathbf{y} \in U$ will converge to $\mathbf{x}$. Then, by virtue of Theorem 1, we have $\mathbf{y}'A\ \mathbf{y} > \mathbf{x}'A\ \mathbf{x}$ for all $\mathbf{y} \in U \setminus \{\mathbf{x}\}$ which shows that $\mathbf{x}$ is a strict local solution for (10).

On the other hand, suppose that $\mathbf{x}$ is a strict local solution of (10), and let $S = \sigma(\mathbf{x})$. By Lemma 1, the function $\mathcal{L} : \mathrm{int}(\Delta_S) \to \mathbb{R}$ defined as $\mathcal{L}(\mathbf{y}) = \pi(\mathbf{x}) - \pi(\mathbf{y})$ is clearly nonnegative in $\mathrm{int}(\Delta_S)$ and it vanishes only when $\mathbf{y} = \mathbf{x}$. Furthermore, $\dot{\mathcal{L}} \leq 0$ by Theorem 1 and, again from Lemma 1, $\dot{\mathcal{L}} < 0$ in $\mathrm{int}(\Delta_S) \setminus \{\mathbf{x}\}$, as $\mathbf{x}$ is the only stationary point in $\mathrm{int}(\Delta_S)$. This means that $\mathcal{L}$ is a strict Liapunov function for any payoff-monotonic dynamics, and hence $\mathbf{x}$ is asymptotically stable; see, e.g., [1, 4]. □

The results presented in this section show that payoff-monotonic dynamics can be usefully employed to find (local) solutions of standard quadratic programs and hence (according to our formulation) data clusters. Indeed, let $G = (V, E, w)$ be a similarity graph and let $A_G$ denote its similarity matrix. By using

$$A = A_G \tag{12}$$

as the payoff matrix, any payoff-monotonic dynamics, starting from an arbitrary initial state, will eventually be attracted with probability 1 by the nearest asymptotically stable point which. By virtue of Theorem 2, this will be a strict local solution of program (2), and hence it will induce a cluster of vertices. In particular, the support of the converged solution will provide the cluster vertices, and the corresponding value of the objective function $f_G$ will measure the cluster cohesiveness.

## 5 An Example: Learning Shape Categories

We illustrate the potential of our approach on the problem of the unsupervised learning of shape categories from an image database. To this end, we used two different similarity matrices reported in the literature derived from as many binary shape databases. Our clustering algorithm consists of iteratively finding a (strict) local solution of program (2) and then removing the support of the solution found (i.e., a cluster) from the graph, until all vertices have been clustered. In our simulations, we used the following discrete-time model:

$$x_i(t+1) = \frac{x_i(t)\pi_i(t)}{\sum_{j=1}^{n} x_j(t)\pi_j(t)} \tag{13}$$

which corresponds to a well-known discretization of (8), [22].

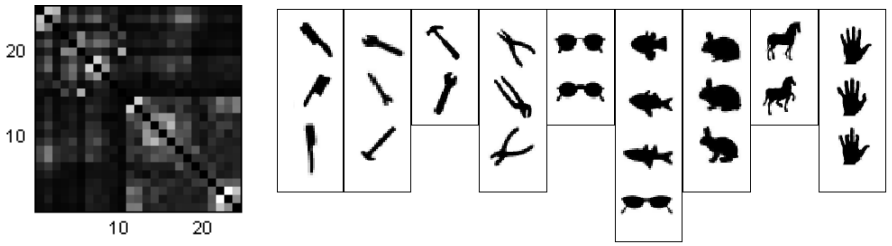



**Fig. 1.** Left: Similarity matrix used in the experiments described in Sect. 5.1. Right: The nine categories discovered by our algorithm

### 5.1 Using Luo et al.'s Similarities

The first dataset we applied our algorithm on was used by Luo et al. [10], who developed a maximum-likelihood framework for graph clustering. The dataset contains 25 different shapes from 9 different classes: brushes, wrenches, pliers, hammers, spectacles, fishes, rabbits, horses, and hands.

In the work described in [10], shapes were abstracted in terms of rooted shock-trees [21], and the similarities between pairs of shapes were computed by using the weighted tree edit distance of the corresponding trees. The resulting similarity matrix (whose entries are taken from [10, Fig. 1] and scaled properly) is shown in Fig. 1 (left). Here, and in the sequel, the gray-levels in the matrix representation are proportional to the similarities: the darker the entries, the weaker the similarities between the corresponding shapes. The order of the entries in the matrix is the same as the one given above.

The results obtained using our clustering algorithm on this dataset are shown in Fig. 1 (right). The algorithm discovered (correctly) 9 classes, and made only 3 mistakes by placing a pair of glasses in the fish class, and by swapping two elements from the hammer and wrench classes. These results are substantially better than those reported by Luo et al. in [10]. Indeed, they were able to partition the database into 7 (instead of 9) classes, and made more classification errors, by placing two fishes in the spectacle class, a hammer in the wrench classes, and by putting together in a quite inhomogeneous cluster, a hammer, a fish, a rabbit, and the two horses.

### 5.2 Using Klein et al.'s Similarities

The second series of experiments was conducted on the database used by Klein et al. in [9], which contains 36 binary shapes from 6 different categories: fishes, wrenches, planes, "greebles," rabbits, and hands. Each category contains 6 different shapes.

They used an unrooted and ordered shock-tree representation for shapes. To measure the distance between two shapes, they first compared each path in one shock graph to each path in the other, computing a cost of deforming one to the other, and then used these costs to compute the edit-distance. Fig. 2 (left)

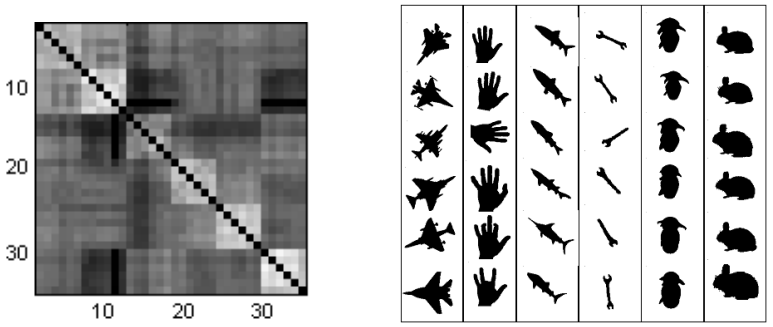



**Fig. 2.** Left: Similarity matrix used in the experiments described in Sect. 5.2. Right: The six categories discovered by our algorithm

shows the similarity matrix obtained by transforming the distances presented in [9, Table 1] into affinities using a customary exponential transformation. The order of the entries is the same as the one given above.

Fig. 2 (right) shows the results obtained using our algorithm, which partitioned the dataset into the right number of classes and made no classification error.

## 6 Conclusions

In this paper, we have proposed a standard quadratic program formulation for the pairwise data clustering problem. We have then introduced a wide family of evolutionary game dynamic equations, known as *payoff-monotonic* dynamics, and have shown how their properties make any member of this family a potential heuristic for solving standard quadratic programs and, in particular, pairwise clustering problems. We have demonstrated the potential of the approach for the problem of the unsupervised learning of shape categories. Experiments with two different similarity matrices (and image databases) reported in the machine vision literature have been conducted, and the results have shown the effectiveness of our approach. Applications of this framework to other (low-level) vision tasks such as intensity and color image segmentation problems are presented in [13]. Finally, in [14] we describe an extension of our framework to hierarchical clustering problems.

## References

[1] N. P. Bhatia and G. P. Szegö. *Stability Theory of Dynamical Systems.* Springer-Verlag, Berlin, 1970. 209

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* J. Wiley & Sons, New York, 2001. 201

[3] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998. 202, 207

[4] M. W. Hirsch and S. Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, New York, 1974. 209

[5] J. Hofbauer. Imitation dynamics for games. Collegium Budapest, preprint, 1995. 206, 207

[6] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, Cambridge, UK, 1998. 202, 204, 206, 207

[7] T. Hofmann and J. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(1):1–14, 1997. 201, 202

[8] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, 1988. 201, 204

[9] P. N. Klein, T. B. Sebastian, and B. B. Kimia. Shape matching using edit-distance: An implementation. In *Proc. 11th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA)*, pages 781–190, 2001. 210, 211

[10] B. Luo, A. Robles-Kelly, A. Torsello, R. C. Wilson, and E. R. Hancock. A probabilistic framework for graph clustering. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2001. 202, 210

[11] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge University Press, Cambridge, UK, 1982. 202

[12] T. S. Motzkin and E. G. Straus. Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.*, 17:533–540, 1965. 202, 203

[13] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 1, pages 145–152, 2003. 202, 203, 211

[14] M. Pavan and M. Pelillo. Dominant sets and hierarchical clustering. In *Proc. IEEE Int. Conf. Computer Vision*, 2003 (to appear). 211

[15] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11(8):2023–2045, 1999. 202

[16] M. Pelillo. Matching free trees, maximal cliques, and monotone game dynamics. *IEEE Trans. Pattern Anal. Machine Intell.*, 24(11):1535–1541, 2002. 202

[17] P. Perona and W. Freeman. A factorization approach to grouping. In H. Burkhardt and B. Neumann, editors, *Computer Vision—ECCV'98*, pages 655–670. Springer-Verlag, Berlin, 1998. 202, 203, 204

[18] L. Samuelson. *Evolutionary Games and Equilibrium Selection*. MIT Press, Cambridge, MA, 1997. 202

[19] S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. *Computer Vision and Image Understanding*, 71(1):110–136, 1998. 202, 203, 204

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Machine Intell.*, 22(8):888–905, 2000. 202, 203

[21] K. Siddiqi, A. Shokoufaundeh, S. J. Dickinson, and S. W. Zucker. Shock graphs and shape matching. *Int. J. Computer Vision*, 35(1):1999, 1999. 210

[22] J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995. 202, 204, 205, 206, 207, 209

[23] S. W. Zucker, A. Dobbins, and L. Iverson. Two stages of curve detection suggest two styles of visual computation. *Neural Computation*, 1:68–81, 1989. 202

# Abduction in Classification Tasks

Maurizio Atzori, Paolo Mancarella, and Franco Turini

Dipartimento di Informatica
University of Pisa, Italy
{atzori,paolo,turini}@di.unipi.it

**Abstract.** The aim of this paper is to show how abduction can be used in classification tasks when we deal with incomplete data. Some classifiers, even if based on decision tree induction like C4.5 [1], produce as output a set of rules in order to classify new given examples. Most of these rule-based classifiers make the assumption that at classification time we can know all about new given examples. Probabilistic approaches make rule-based classifiers able to get the most probable class, on the basis of the frequency of the missing attribute in the training set [2]. This kind of assumption sometimes leads to wrong classifications. We present an abductive approach to (help to) choose which classification rule to apply when a new example with missing information needs to be classified, using knowledge about the domain.

## 1 Introduction

Due to the availability of large amounts of data, easily collected and stored via computer systems, the field of so-called data mining is gaining momentum. Several important results have been obtained in the context of specific algorithms, in applying the techniques in several application fields, and in designing suitable environments in which the data mining step can be embedded. Such environments support the phases that come before (e.g. cleaning the data) and the ones that come after (e.g. visualization of results), and attempt also at providing a context in which one can process the results of the data mining step in order to answer higher level questions than the ones directly provided by the computed data mining model.

For example, extracting the association rules from a set of supermarket transactions is very useful, and can answer basic questions like "which are the items that induce a buying attitude towards other items?", but it would be even more interesting answering the question "did the new layout of the store influence the buying attitude?". Answering the last question requires taking the association rules computed with respect to the old layout and comparing them with the ones computed according to the new ones, possibly taking into account the rules underneath the new design.

In brief, we believe that the results of data mining algorithms may be the input to a reasoning environment, where high level questions can be answered by the exploitation of both the results of data mining steps and some "domain knowledge". A general environment suited for this endeavor is described in [3].

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 213–224, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In this paper we concentrate on a very focussed goal: proving that a careful representation of the results of a data mining algorithm (a decision or classification tree in this case), a careful representation of extra domain knowledge (constraints in the case at hand), and a careful choice of a reasoning technique (abduction in the case at hand) can substantially improve the behavior of the extracted model (the activity of classification in the case).

Classification trees are one of the main models extracted from web logs data in the *Clickworld* Project, that aims at improving the management of web sites through knowledge discovery means.

In Sect. 2 we sketch some basic concepts on data mining and decision trees and on abductive reasoning, and we will set up the notations and terminology used throughout the paper. In Sect. 3 we formalize the concept of classification in decision trees and formally show how it can be viewed as an abductive problem. In Sect. 4 we show that the abductive view of the classification tasks suggests useful extensions of the latter by exploiting domain-specific knowledge represented as integrity constraints. Finally, in Sect. 5 we draw some lines of future research on the subject.

## 2 Preliminaries

In order to understand the idea that we are going to describe later in this paper, we briefly review some background on data mining and decision trees and on abduction.

### 2.1 Data Mining

Data mining can be defined as the process of finding correlations or patterns among dozens of fields in large relational databases. It is an essential step of the knowledge discovery process where intelligent methods are applied in order to extract data patterns from very large databases [4]. In particular, we are interested in the classification task, that is predicting categorical labels given some examples.

Classification of data requires two sequential steps: the first one consists in building a model that describes a given set of examples by associating a class label to each of them; the second one concerns using the model to classify new examples (i.e. predict the categorical label). The only model we are interested in in this paper is the one of the *decision trees* that we are going to briefly describe.

**Decision Trees.** A decision tree is a tree structure in which each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represent classes. Decision tree induction consists in building such a tree from a training set of examples and then using it (following a path from the root to a leaf) to classify new examples given their attribute values. Because of their structure, it is natural to transform decision trees into classification rules, that can be easily inserted into a reasoning framework. Notice that some machine

learning tools, such as C4.5 [1], already includes a class rulesets generator. In the sequel we will see how class rulesets can be embedded into an abductive reasoning framework which will allow us, in some cases, to better classify new examples in presence of external information, such as specific domain knowledge.

Let us now set up the main notations and terminologies we will use throughout the paper as far as decision trees are concerned. Let $\mathcal{A}$ be a set of attribute names and $\mathcal{C}$ be a set of classes (possible classifications). For simplicity, we assume that each attribute can be assigned a value over a finite set of values $\mathcal{V}$. An *example* $e$ is a set of attribute/values pairs

$$e = \{a_1 = v_1, \ldots, a_n = v_n\}$$

where all the $a_i$s are distinct attribute names.

**Definition 1.** *A* decision tree $T$ *over* $\mathcal{A}$ *and* $\mathcal{C}$ *is a tree such that:*

*(i) each non-leaf node is labelled by an attribute* $a \in \mathcal{A}$*;*
*(ii) each leaf node is labelled by a class* $c \in \mathcal{C}$*;*
*(iii) each branch is labelled by a value* $v \in \mathcal{V}$*;*
*(iv) the values labelling all the branches exiting from a given node are all distinct;*
*(v) the labels of a path are all distinct.*

Notice that *(v)* formalizes the fact that, in each path, only one test can be performed on each attribute.

*Example 1.* Let us consider a very well known example, taken from [1]. Given a training set of examples which represent some situations, in terms of weather conditions, in which it is or it is not the case that playing tennis is a good idea, a decision tree is built which can be used to classify further examples as good candidates for playing tennis (class $Yes$) and bad candidates to play tennis (class $No$). Table 1 shows the original training set, given as a relational table over the attributes $\{Outlook, Temperature, Humidity, Wind\}$. The last column of the table represents the classification of each row.

Using standard decision trees inductive algorithms (e.g., ID3), we may obtain the decision tree in Fig. 1 from the above training set. As we have already pointed out, each internal node represents a test on a single attribute and each branch represents the outcome of the test. A path in the decision tree represents the set of attribute/values pairs that an example should exhibit in order to be classified as an example of the class labelled by the leaf node. For instance, given the above tree, the example $\{Overlook = Sunny, Humidity = Low\}$ is classified as $Yes$, whereas the example $\{Overlook = Sunny, Humidity = High\}$ is classified as $No$. Notice that not all the attribute values have to be specified in order to find a classification of an example. On the other hand, if an example is too under-specified, it may lead to different, possibly incompatible, classifications. For instance, the example $\{Overlook = Sunny\}$ can be classified both as $Yes$ or $No$, following the two left-most branches of the tree. It is also worth noticing that the decision tree may not consider all the attributes given in the training set. For instance, the attribute $Temperature$ is not taken into account at all in the previous decision tree.

**Table 1.** Training set of examples on attributes *Overlook, Temperature, Humidity, Wind*. The values *Yes* and *No* represent the concept to learn

| Outlook | Temperature | Humidity | Wind | Class |
|---|---|---|---|---|
| Sunny | Hot | High | Weak | No |
| Sunny | Hot | High | Strong | No |
| Overcast | Hot | High | Weak | Yes |
| Rainy | Mild | High | Weak | Yes |
| Rainy | Cool | Low | Weak | Yes |
| Rainy | Cool | Low | Strong | No |
| Overcast | Cool | Low | Strong | Yes |
| Sunny | Mild | High | Weak | No |
| Sunny | Cool | Low | Weak | Yes |
| Rainy | Mild | Low | Weak | Yes |
| Sunny | Mild | Low | Strong | Yes |
| Overcast | Mild | High | Strong | Yes |
| Overcast | Hot | Low | Weak | Yes |
| Rainy | Mild | High | Strong | No |

### 2.2 Abductive Logic Programming

Abduction is a particular form of reasoning introduced by the philosopher Peirce [5] as a form of synthetic reasoning which infers the case from a rule and a result, i.e. from the fact that $A$ implies $B$ and the observation $B$, we can abduce $A$. Abduction can be viewed as the probational adoption of an hypothesis as an explanation for observed facts, according to known laws. Obviously, as Peirce noticed, "it is a weak kind of inference, because we cannot say that we believe in the truth of the explanation, but only that it may be true".

In recent years, abduction has been widely studied and adopted in the context of logic programming [6], starting from the work [7]. In this paper, we basically

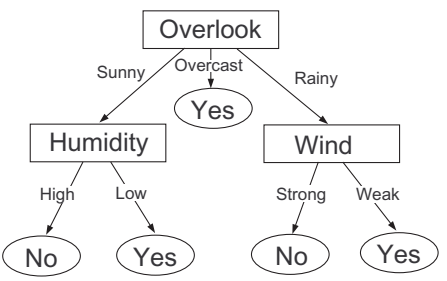



**Fig. 1.** A well-know example of decision tree, obtained from the training set of Table 1

adopt the definitions and terminologies of [8, 9], that we are going to briefly review next.

An Abductive Logic Programming (ALP) framework is defined as a triple $\langle P, A, Ic\rangle$ consisting of a logic program, $P$, a set of ground abducible atoms $A$ and a set of logic formulas $Ic$, called the integrity constraints. The atoms in $A$ are the possible abductive hypotheses which can be assumed in order to explain a given observation in the context of $P$, provided that these assumptions are consistent with the integrity constraints in $Ic$. In many cases, it is convenient to define $A$ as a set of (abducible) predicate symbols, with the intended meaning that each ground atom whose predicate symbol is in $A$ is a possible hypothesis.

Given an abductive framework as before, the general definition of abductive explanation is given as follows.

**Definition 2.** *Let $\langle P, A, Ic\rangle$ be an abductive framework and let $G$ be a goal. Then an* abductive explanation *for $G$ is a set $\Delta \subseteq A$ of ground abducible atoms such that:*

- $P \cup \Delta \models G$
- $P \cup \Delta \cup Ic$ *is consistent.*

*Example 2.* Let $\langle P, A, Ic\rangle$ be an abductive framework where:

- $P$ is the logic program given by the three rules: $p \leftarrow a \quad p \leftarrow b \quad q \leftarrow c$
- $A = \{a, b, c\}$
- $Ic = \{\}$.

Then, there are three abductive explanations for the goal[1] $(p, q)$ given by $\Delta_1 = \{a, c\}$, $\Delta_2 = \{b, c\}$ and $\Delta_3 = \{a, b, c\}$.

Consider now the abductive framework $\langle P, A, Ic'\rangle$, where $P$ and $A$ are as before and $Ic$ contain the formula $\neg(a, c)$. In this new framework, there is only one explanation for the goal $p, q$, which is $\Delta_2$ above, since the second potential explanation is inconsistent with the integrity constraint.

The given notion of abductive explanation can be easily generalized to the notion of abductive explanation given an initial set of abducibles $\Delta_0$.

**Definition 3.** *Let $\langle P, Ab, Ic\rangle$ be an abductive framework, $\Delta_0$ be a set of abducibles and $G$ be a goal. We say that $\Delta$ is an abductive explanation for $G$* given *$\Delta_0$ if $\Delta_0 \cup \Delta$ is an abductive explanation for $G$.*

Notice that this implies that the given set of abducibles $\Delta_0$ must be consistent with the integrity constraints $Ic$. In some cases, we may be interested in *minimality* of abductive explanations.

**Definition 4.** *Let $\langle P, Ab, Ic\rangle$ be an abductive framework, $\Delta_0$ be a set of abducibles and $G$ be a goal. We say that $\Delta$ is a $\Delta_0-$minimal explanation for $G$ if $\Delta$ is an explanation for $G$ given $\Delta_0$ and for no proper subset $\Delta'$ of $\Delta$ ($\Delta' \subset \Delta$), $\Delta'$ is an explanation for $G$ given $\Delta_0$.*

[1] As in standard logic programming, "," in rule bodies and goals denotes logical conjunction.

For instance, in Example 2 with $Ic = \{\}$, $\Delta_0$ is the empty set (we do not have an initial set of "pre-abduced" abducibles) and both $\Delta_1$ and $\Delta_2$ are $\Delta_0-$minimal explanations for $(p, q)$ while $\Delta_3$ is not (in fact $\Delta_1 \subset \Delta_3$).

The notion of abduction as described above has a natural computational counterpart within a proper extension of the standard SLD-based computational engine of logic programming. In particular, when integrity constraints are in the form of *denials*, i.e. in the form $\neg(p_1, \ldots, p_n)$, the so called KM-proof procedure can be successfully adopted to compute (minimal) abductive explanations. Due to lack of space we omit the details of the definition of the KM-proof procedure which can be found in [8, 9].

## 3 Classification as an Abductive Problem

In this Section we show how the classification task using decision trees can be directly seen as an abductive problem. To do this, we first need to spend some words on the notation and then formally define what decision tree classification means and on the transformation of decision trees into abductive logic programs.

Given a tree $T$ and a path $\pi$ in $T$, we denote by $Attr(\pi)$ the example $\{a_1 = v_1, \ldots, a_k = v_k\}$ where each $a_i$ is a non-leaf node in $T$ and each $v_i$ is the attribute value labelling the branch exiting from $a_i$. Moreover, we denote by $Class(\pi)$ the class labelling the leaf node of $\pi$.

The following defines the concept of classifying an example by means of a decision tree in case of missing information. The usual definition of classifying an example is given as a special case, in which there are no missing information.

**Definition 5.** *Let $T$ be a decision tree,* e *be an example and* c *be a class. We say that the example* e **may be** *classified as* c *by $T$ via $\delta$, denoted by $T \overset{c,\delta}{\Longrightarrow}$* e *if there exists a path $\pi$ in $T$ with leaf node $Class(\pi) = c$ such that $e \cup Attr(\pi)$ is an example and $\delta = Attr(\pi) \setminus e$.*
*If $T \overset{c,\{\}}{\Longrightarrow}$* e *we say that the example* e **is** *classified as* c *by $T$, and we simply write $T \overset{c}{\Longrightarrow}$* e

Notice that, in $T \overset{c,\delta}{\Longrightarrow} e$, the condition that $e \cup Attr(\pi)$ must be an example ensures that the attribute values of $e$ are compatible with the set of tests represented by $\pi$. In other words, if for some attribute $a$, $a = v \in e$ and $a = v' \in Attr(\pi)$, then $v = v'$. Moreover, notice that $\delta$ represents the extension to the example $e$ which is needed in order to classify it as $c$ from the chosen path.

*Example 3.* Let $T$ be the decision tree of Example 1. We have:

- $T \overset{Yes}{\Longrightarrow} \{Overlook = Sunny, Wind = Strong, Humidity = Low\}$
- $T \overset{No,\delta}{\Longrightarrow} \{Overlook = Sunny\}$, where $\delta = \{Humidity = High\}$

Now, let us explain how to transform a decision tree into an abductive logic program. Let $T$ be a decision tree and $\pi$ be a path in $T$. The rule $r_\pi$ associated with $\pi$ is the Horn clause $c \leftarrow a_1(v_1), \ldots, a_n(v_n)$ such that:

– $Class(\pi) = c$, and
– $Attr(\pi) = \{a_1 = v_1, \ldots, a_n = v_n\}$

Notice that attribute names are viewed as unary predicate symbols, and that an attribute/value pair $a = v$ is mapped into an atom $a(v)$. Moreover, by *(v)* in Def. 1, given a path $\pi$, $Attr(\pi)$ is an example. The Horn clause program $P_T$ associated with a decision tree $T$ is the set of rules

$$P_T = \{r_\pi \mid \pi \text{ is a path in } T\}.$$

Finally, we associate with $T$ the set $IC_T$ of *canonical* integrity constraint containing a denial

$$\leftarrow a(x), a(y), x \neq y$$

for each attribute $a \in \mathcal{A}$.

As we will see next, attribute names are viewed as abducible predicates, and sets of abducibles will represent possible classifications. The integrity constraints we have just defined ensure that in each consistent set of abducibles, an attribute occurs at most once (cfr. case (v) of Def. 1).

Given an example $e = \{a_1 = v_1, \ldots, a_k = v_k\}$, let $\Delta_e$ be the set of atoms

$$\Delta_e = \{a_1(v_1), \ldots, a_k(v_k)\}.$$

We have set up all the ingredients which are needed to associate an abductive framework to a decision tree $T$.

**Definition 6.** *Given a decision tree $T$, the abductive framework $AB_T$ associated with $T$ is the triple*

$$AB_T = \langle P_T, \mathcal{A}, IC_T \rangle$$

*where $P_T$ is the Horn program associated with $T$, $IC_T$ is the set of canonical integrity constraints associated with $T$ and $\mathcal{A}$ is the set of attribute names.*

The following theorem formalizes the correspondence between classifications in $T$ and abductive explanations in $AB_T$.

**Theorem 1.** *Let $T$ be a decision tree and $AB_T$ be the corresponding abductive framework. Let also* e *be an example. Then $\Delta$ is a $\Delta_e-$minimal explanation for* c *with respect to $AB_T$ if and only if for some path $\pi$ in $T$ we have $T \overset{c,\delta}{\Longrightarrow} e$ and $\Delta = \Delta_\delta$.*

*Proof.*
($\Longleftarrow$) Assume that $T \overset{c,\delta}{\Longrightarrow} e$. Then, by Def. 5, for some path $\pi$ in $T$, $e \cup Attr(\pi)$ is an example and $\delta = Attr(\pi) \setminus e$. Consider the rule $r_\pi$

$$c \leftarrow a_1(v_1), \ldots, a_n(v_n).$$

and let $body(r_\pi) = \{a_1(v_1), \ldots, a_n(v_n)\}$. It is clear that $body(r_\pi) = \Delta_{Attr(\pi)}$ and $P_T \cup body(r_\pi) \models c$. Let $\Delta = body(r_\pi) \setminus \Delta_e$: clearly, $P_T \cup \Delta_e \cup \Delta \models c$ and $\Delta =$

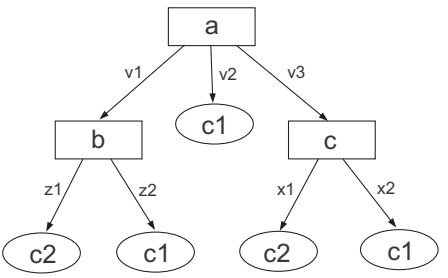



**Fig. 2.** A simple example of decision tree

$\Delta_\delta$. Moreover, since $e \cup Attr(\pi)$ is an example and $(\Delta_e \cup \Delta) \subseteq \Delta_{e \cup Attr(\pi)}$, $P_T \cup \Delta_e \cup \Delta \cup IC_T$ is consistent. Hence, $\Delta$ is an explanation for $c$ given $\Delta_e$. By construction, $\Delta$ is also a $\Delta_e-$minimal explanation.
($\Longrightarrow$) Assume $\Delta$ is a $\Delta_e-$minimal explanation for $c$. Clearly, $\Delta \cap \Delta_e = \{\}$ by minimality of $\Delta$. Since $P_T \cup \Delta_e \cup \Delta \models c$, by construction of $P_T$ there exists a path $\pi$ in $T$ such that $r_\pi$ is a rule of the form

$$c \leftarrow a_1(v_1), \ldots, a_n(v_n)$$

and $\{a_1(v_1), \ldots, a_n(v_n)\} \subseteq \Delta_e \cup \Delta$.

Let $\delta = Attr(\pi) \setminus e$. By construction and minimality of $\Delta$, we have $\Delta = \Delta_\delta = \{a_1(v_1), \ldots, a_n(v_n)\} \setminus \Delta_e$. Finally, it is clear that $e \cup Attr(\pi)$ is an example, by the consistency of $P_T \cup \Delta_e \cup \Delta \cup IC_T$ and the observation that $\Delta_{e \cup Attr(\pi)} = \Delta_e \cup \Delta$. □

*Example 4.* Let us consider the following decision tree (which is nothing but the decision tree of Example 1 using symbolic names and symbolic values). In the abductive framework $AB_T = \langle P_T, \mathcal{A}, IC_T \rangle$ associated with $T$ we have:

- $P_T$ is the following set of rules:
  $c1 \leftarrow a(v2).$ $\quad c2 \leftarrow b(z1), a(v1).$
  $c1 \leftarrow b(z2), a(v1).$ $\quad c2 \leftarrow c(x1), a(v3).$
  $c1 \leftarrow c(x2), a(v3).$
- $IC_T$ is the following set of canonical integrity constraints:
  $\leftarrow a(x), a(y), x \neq y.$ $\quad \leftarrow b(x), b(y), x \neq y.$ $\quad \leftarrow c(x), c(y), x \neq y.$
- $\mathcal{A} = \{a, b, c\}$.

Let us consider the example $e = \{a = v1, b = z1\}$ and the corresponding initial set of abducibles $\Delta_e = \{a(v1), b(z1)\}$. Clearly $\Delta = \{\}$ is a $\Delta_e-$minimal explanation for $c2$, and indeed it corresponds to the leftmost path in the tree. Notice that also $\Delta' = \{c(x1)\}$ is an explanation for $c2$ given $\Delta_e$, but it is not $\Delta_e-$minimal and indeed it does not correspond to any path in the tree matching the attribute/values pairs given in $e$.

## 4 Improving Classification Exploiting Domain Specific Knowledge

The abductive view of classification given in the previous Section can be seen as an alternative, though equivalent, way of performing classification given some decision tree. It is worth noting that the abductive reasoning required in this alternative view is very limited and does not fully exploit the potential power of abductive reasoning in logic programming as sketched in Sect. 2.2. In particular, the transformation requires the use of a pre-defined, canonical set of integrity constraints that simply avoid explanation to contain different values for the same attribute. We show in this Section that one way to exploit abductive reasoning is to add *domain specific* knowledge in order to improve the classification task. Domain specific knowledge may be taken into account during classification in many ways, e.g. to rule out some classifications or to prefer one particular classification over another. Up to our knowledge, taking into account domain specific knowledge in standard decision-tree based classification algorithms may be not straightforward and may require substantial modifications of these algorithms. On the other hand, abductive frameworks, and existing concrete implementations of them, are already equipped with mechanisms that can be directly exploited to represent and handle domain specific knowledge.

In this paper we exploit integrity constraints (beyond the canonical ones) in abductive frameworks as a way to express domain specific knowledge. Let us show some examples.

*Example 5.* Consider the decision tree $T$ of Example 4. As we have already pointed out, the attributes which label the internal nodes of decision trees may not be all the attributes that examples are equipped with. In the current example, assume that the original set of attributes contained also an attribute $d$, beyond the attributes $\{a, b, c\}$ occurring in the tree. Assume now that an example is given $e = \{d = w1\}$: it is clear that, since $d$ does not occur in the decision tree, any classification of $e$ can be done using $T$. However, assume that we have extra knowledge on the domain at hand, expressing the fact that the value $w1$ for the attribute $d$ is incompatible both with the value $v1$ of the attribute $a$ and with the value $x1$ of the attribute $c$. It is easy to see that this extra knowledge can be used to classify the example $e$ as belonging to class $c1$. In the abductive framework corresponding to $T$ the extra knowledge can be easily coded by adding the following integrity constraints:

$$\neg(d(w1), a(v1)) \qquad \neg(d(w1), c(x1))$$

If we consider now the full abductive framework $\langle P, \mathcal{A}', Ic\rangle$, where $P$ is the program obtained as in Example 4, $\mathcal{A}' = \{a, b, c, d\}$ and $Ic$ are the integrity constraints we have just shown, we observe that $c2$ has no abductive explanation given $\Delta_e = \{d(w1)\}$, whereas the goal $c1$ has two $\Delta_e-$minimal explanations given $\Delta_e$, namely $\Delta_1 = \{a(v2)\}$ and $\Delta_2 = \{a(v3), c(x2)\}$. Notice the correspondence between these two solutions and the two right-most paths with $c1$ as

the leaf node. From a computational point of view the two explanations can be computed using, e.g., the KM-proof procedure. The consistency checking phase of the proof procedure rules out the two potential $\Delta_e-$minimal explanations $\{b(z1), a(v1)\}$ and $\{a(v3), c(x1)\}$ for the goal $c2$. Indeed, the first one is inconsistent with the integrity constraint $\neg(d(w1), a(v1))$, and the second one is inconsistent with the integrity constraint $\neg(d(w1), c(x1))$.

As the previous example points out, integrity constraints can be used to add knowledge relating the values of the various attributes, including those which do not occur in the original decision tree. This can help and improve the classification task. The next example shows that integrity constraints may add knowledge which is relevant to the attributes already occurring in the decision tree, although it is not explicit in the decision tree itself.

*Example 6.* Let us consider again the decision tree of Example 1. Assume that, as it is often the case, whenever there is strong wind the humidity is not high:

$$\neg(Humidity(High), Wind(Strong)).$$

It is important to point out that this kind of knowledge may not be implicit in the training set from which the original decision tree was built. Actually, the examples in the training set may even contradict this knowledge (see, e.g., Table 1). Indeed, this knowledge may arise from knowledge sources different from the ones which provide the training set (in this particular example this knowledge may be associated with typical weather forecast knowledge bases). Assume now that we want to classify an example described simply as $e = \{Overlook = Sunny, Wind = Strong\}$. In the corresponding abductive framework, given the initial set $\Delta_e = \{Overlook(Sunny), Wind(Strong)\}$, the classification $Yes$ has an abductive explanation $\Delta_1 = \{Humidity(Low)\}$ and the classification $No$ has an abductive explanation $\Delta_2 = \{Humidity(High)\}$. If we consider now the above integrity constraint, the abductive explanation $\Delta_2$ is ruled out, due to the fact that the full explanation given by $\Delta_e \cup \Delta_2 = \{Overlook(Sunny), Wind(Strong), Humidity(High)\}$ is inconsistent. Thus, by adopting the very same computational mechanism we obtain a correct classification with respect to the extra domain specific knowledge.

In many cases, decision trees can be equipped with probability measures on the outcome of attribute/values test. In other words, each branch of the tree is labelled both by a value corresponding to the attribute labelling the parent node, as well as with a probability measure which indicates how likely is that an observation exhibits this value for the given attribute. This kind of probability information is clearly useful when we try to classify new examples with missing attribute information. Even in this case, extra domain knowledge can be taken into account in the classification task in order to dynamically get better probability measures on possible classification of new under-specified examples. To show this let us consider a very simple example.

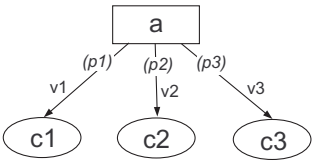



**Fig. 3.** A simple example of a decision tree with probabilities associated to each branch

*Example 7.* Let us consider the simple decision tree in Fig. 3. There is only one internal node, the root, which represents a test on a 3-valued attribute named $a$. Probabilities are represented between parenthesis. The classifications rules extracted from the above tree are the following:

$$c1 \leftarrow a(v1) \qquad c2 \leftarrow a(v2) \qquad c3 \leftarrow a(v3).$$

Notice that we could have more information about a new observation than the $a$ attribute only, but the tree-based classification (and its associated class ruleset) will use only information about $a$. But what happens if we know nothing about $a$? If we try to classify a new example with no information about $a$ we get as output the class with highest probability. Again, using integrity constraints in the corresponding abductive framework which makes explicit extra information about the domain, we can cut some branches during the classification task, thus obtaining better classifications and also dynamically improving probability values. For instance, let this domain knowledge be represented by the integrity constraint $\neg(a(v1), b(z2))$. In this case, if we are asked to classify the example $\{b = z2\}$ we will discard the classification $c1$ whose abductive explanation $\Delta_1 = \{a(v1)\}$ is inconsistent with the integrity constraint. On the other hand, we would compute the classifications $c2$ and $c3$ with the corresponding explanations $\Delta_2 = \{a(v2)\}$ and $\Delta_3 = \{a(v3)\}$. We can also dynamically re-compute the associated probabilities which will be $\frac{p2}{p2+p3}$ as far as $c2$ is concerned, and $\frac{p3}{p2+p3}$ as far as $c3$ is concerned.

## 5 Conclusions

Starting from the idea that framing the data mining step into a logic framework provides us with the possibility of exploiting also domain knowledge in the knowledge extraction and exploitation process, we showed that the result of a classification tree can be improved if the tree is not simply traversed, but it is visited in abductive mode. Indeed, the basic observation is that the application of a classification tree to a new observation can be viewed as a straightforward abductive computation, as soon as the tree is represented as a collection of rules in the obvious way. This observation opens up a world of possibilities, since the abductive computations can become very sophisticated, and they can take into

account several types of knowledge. In this paper we restricted our attention to domain knowledge represented as simple constraints involving equalities, conjunction and negation. However, even with this limited power, we could show examples in which the classification task was radically improved. We are confident that more sophisticated use of knowledge can lead us to much better improvements.

Several frameworks in the context of logic programming have been already developed in order to integrate induction and abduction (see, e.g., [10, 11, 12]).

The framework we have outlined is anyway the first one, as far as we know, in which the inductive task performed by external machine learning algorithms is not influenced by abductive procedures, but post-processed through abduction reasoning in order to take into account the domain knowledge and to improve in some cases the classification task itself.

## Acknowledgments

This work has been carried out within the project *Clickworld* (MIUR S585-P).

## References

[1] Quinlan, J. R.: C4.5: programs for machine learning. Morgan Kaufmann (1993) 213, 215

[2] Quinlan, J. R.: Unknown attribute values in induction. In: Proc. of the Sixth International Machine Learning Workshop, Morgan Kaufmann (1989) 164–168 213

[3] Giannotti, F., Manco, G., Pedreschi, D., Turini, F.: Experiences with a logic based knowledge discovery support environment (Springer-Verlag 1999) 213

[4] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2000) 214

[5] Peirce, C. S.: (Collected papers of Charles Sanders Peirce) Vol. 2, Hartshorn et al. eds., Harvard Univesity Press. 216

[6] Kakas, A., Kowalski, R., Toni, F.: Abductive logic programming. Journal of Logic and Computation **2** (1993) 719–770 216

[7] Eshgi, K., Kowalski, R.: Abduction compared with negation by failure. In Levi, G., Martelli, M., eds.: Proc. of the 1989 International Conference on Logic Programming, MIT Press (1989) 234–254 216

[8] Kakas, A., Mancarella, P.: Generalized stable models: a semantics for abduction. In: Proc. of 9th European Conference on Artificial Intelligence, Pitman (1990) 385–391 217, 218

[9] Kakas, A., Michael, A., Mourlas, C.: Aclp: Abductive constraint logic programming. Journal of Logic Programming **44** (2000) 129–177 217, 218

[10] Lamma, E., Mello, P., Milano, M., Riguzzi, F.: Integrating induction and abduction in logic programming. Information Science **116** (1999) 25–54 224

[11] Kakas, A., Riguzzi, F.: Abductive concept learning. New Generation Computing **18** (2000) 224

[12] Flach, P. A., Kakas, A. C., eds.: Abduction and Induction: Essays on their relation and integration. Kluwer Academic Publishers (2000) 224

# MQL: An Algebraic Query Language for Knowledge Discovery

Miriam Baglioni and Franco Turini

Department of Computer Scienze, University of Pisa
Via F. Buonarroti 2, 56125 Pisa Italy
{baglioni,turini}@di.unipi.it

**Abstract.** MQL is a system supporting the process of Knowledge Discovery. The central step of knowledge discovery, i.e. the application and combination of data mining steps, is expressed via queries written in an algebraic query language. The query processing engine exploits an XML based representation of queries and data mining models to favor the interoperability of different data mining tools and the expandibility of the system.

## 1 Introduction

Knowledge Discovery in Databases is a process that includes several steps: data pre-processing, model extraction etc. Ad hoc tools are available but the definition and construction of a general framework supporting the whole process is still matter of research. As stated in[9], the current state of the art in KDD is very similar to what was under way in the DBMS field in the 60's, when each application had to be built by writing a specific program for each data extraction. The definition of SQL allowed us to overcome this limitation. By analogy, we should look at the KDD process as a query process. MQL, the language and system presented here, is designed by considering the KDD process as a query process according to the suggestions given in[9]. The basic points are that a KDD query language must verify the property of closure, and that two classes of objects have to be supported: KDDObjects and KDDQueries. KDDObjects are the models that can be extracted from the data, and KDDQueries are queries returning KDDObjects or basic database objects (records, tuples). In MQL the concept of KDDObject has been extended to include tables and hierarchies, and it is referred to as G_OBJ. This choice allows one to operate, in a uniform way, with the results of KDDQueries, independently from whether they are database objects or KDDObjects. Thanks to hierarchies, it is also possible to extract models at different levels of abstraction. KDDQueries are represented in the language by means of derived operations. Such operations can be nested: the result of an operation can be the input to another operation of compatible type. In this respect MQL verifies the concept of closure in query languages. MQL has been designed as an algebraic language. The semantics is given with respect to the general notion of sequence, and algebraic properties of operators can be proved

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 225–236, 2003.
© Springer-Verlag Berlin Heidelberg 2003

in terms of basic properties of sequences.
The system is implemented by translating the queries into an XML based internal representation that is handled by a query processing engine, named KDDML[1]. The engine has been designed and built with the aim of making the interoperability among data mining algorithms and KDD tools in general easy to obtain. The basic idea is that both queries and models are represented by XML definitions on one side, and that the architecture is open to the addition of new tools. Summing up, MQL is an algebraic query language for knowledge discovery. Complex processes of knowledge extraction, in which different data mining algorithms have to be combined, can be expressed as queries. The system that supports MQL is extensible to include new data mining algorithms and data transformation procedures. The ease of extending the system is provided by the XML based middleware of the system. The paper is organized as follows: Section 2 describes the principal features of the language by means of an example. Section 3 offers an overview of the semantics of MQL and a taste of the kind of algebraic properties that can be proved. Section 4 deals with the implementation of the system and the conclusions include comparisons to other proposals, and plans for future work.

## 2 MQL by Examples

The aim of this section is to present an example of the use of the language MQL, in order to show its expressive power and its capability of combining several KDD techniques. Consider the following problem taken from[9]:
"Generate a classifier trained on a user defined training set with user defined attributes. Find all records in the database wrongly classified using that classifier, and use that set as a training data for another classifier".

Table 1 shows a training set with four attributes and two classes. We are interested in rows with value "false" for the attribute "windy". Under these hypotheses the problem is solved by the following query:

```
begin query
let TS = take all by weather if windy == false;
        AD = CreateTree play from TS C4.5 true true 0.5 1;
        IC = Use AD With weather;
        IM = Misclassified IC
in AM = CreateTree misc from IM C4.5 true true 0.5 1;
end query
```

The `take all by weather if windy==false` operation returns a new table (TS) where each tuple has a "false" value for the field "windy"; if compared to SQL it is similar to `select * from weather where windy=false`. We generate a classifier using TS as a training set. The `CreateTree play from TS C4.5 true true 0.5 1` operation yields a new classifier (AD) using the "C4.5" algorithm[15]. We use AD to classify all tuples in "weather" using the operation `Use AD With weather`, that returns a new table (IC) where every tuple has

**Table 1.** A small training set (weather)

| outlook | temperature | humidity | windy | play |
|---|---|---|---|---|
| sunny | 85 | 85 | false | no |
| sunny | 80 | 90 | true | no |
| overcast | 83 | 86 | false | yes |
| rainy | 70 | 96 | false | yes |
| rainy | 68 | 80 | false | yes |
| rainy | 65 | 70 | true | no |
| overcast | 64 | 65 | true | yes |
| sunny | 72 | 95 | false | no |
| sunny | 69 | 70 | false | yes |
| rainy | 75 | 80 | false | yes |
| sunny | 75 | 70 | true | yes |
| overcast | 72 | 90 | true | yes |
| overcast | 81 | 75 | false | yes |
| rainy | 71 | 91 | true | no |
| overcast | 80 | 74 | false | yes |

**Table 2.** The IM table

| outlook | temperature | humidity | windy | play |
|---|---|---|---|---|
| rainy | 65 | 70 | true | no |
| sunny | 75 | 70 | true | yes |
| rainy | 71 | 91 | true | no |

both observed value and classification value. The `Misclassified IC` operation returns a new table (IM) where every instance is misclassified by AD. The result of this operation is shown in Table 2.

AM is a classifier generated using IM as training data.

## 3 Syntax and Semantics of MQL

The abstract syntax of the language is specified by the signature in Fig. 3. It shows the objects of the language and the operations that can be used to manipulate them. Every operation takes only a specified object type as input and it can be considered as a query associated to its output object type. This feature makes it possible to use the result of an operation as input for another operation of compatible type. You can also combine several types of knowledge extraction operations, generating, for example, a classifier from the instances of one of the clusters obtained from a clustering step. Let us now briefly comment some of the operations that have not been used in the example above:

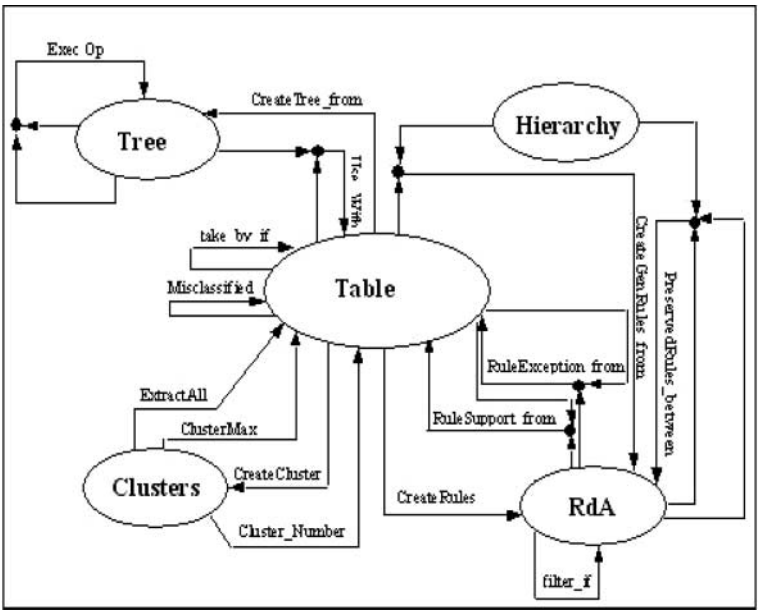



**Fig. 1.** The graph that represents MQL syntax

**Exec Op** , takes two classifiers and returns a new classifier by applying And, Or or Committee to the input. The type of the resulting classifier is specified by **Op**;

**PreserverdRules_between** , takes two sets of association rules - one of them obtained by a set of items, the other one by the same generalised set -, a hierarchy on the items we want to extract association rules from, and an integer. This operation yields a new set of rules that contains every rule that is preserved, with respect to the hierarchy, at the level of abstraction specified by the fourth input parameter;

**CreateGenRules_from** , takes a set of items and a hierarchy built on the items and returns a new set of generalized association rules;

**RuleSupport_from** , takes a set of association rules and a table, T. It returns a new table composed by every instance in T that supports the input rules;

**Cluster_Number** , takes a set of clusters and a cluster identifier "n". It returns a table composed by the instances belonging to the cluster n;

**ExtractAll** , takes a set of clusters and yields a table composed by all the instances belonging to all the clusters stripped of the clustering attribute

The aim of the rest of this section is to provide an overview of the semantics of the language. Every object in the language is associated to a general declaration, called G_DECL, and to a general representation, called G_OBJ. With respect to association rules, for example, G_DECL is the following:

RdA
@IdRule
@Support
@Confidence
@Body
@Head#

The above G_DECL specifies that every object representing an association rule must be composed of five elements.

The G_OBJ associated to the specific set of rules, shown in Table 3, is the following:

RdA#Rule1,0.5,0.5,I1,I6;Rule2,0.5,0.5,(I7,I9),I10;
Rule3,0.5,0.5,I8,(I5,I7);Rule4,0.5,0.5,(I6,I3),(I4,I12)#

This G_OBJ specifies that the type of knowledge is a set of association rules, and that it is composed of four objects. Although each G_OBJ type has its own specific structure, all G_OBJ types are constructed according to the general notion of sequence, that play the role of basic semantic model of our algebra, as much as tables play this role in relational algebra. In fact, every object that represents a specific type is composed of the sequence of elements expressed in the G_DECL for the type itself. You can also see the G_OBJ as composed of a sequence of element, where every element is an object that represents the knowledge type expressed in the G_OBJ. So it is possible to define operations, called elementary operations, to manipulate a sequence of elements independently from the G_OBJ type, that the sequence is associated to. The elementary operations are the following:

***Extraction of a Sequence of Elements from G_OBJ:*** $\varepsilon(Type, Name, G_OBJ)$
The result of this operation is a sequence of elements, that can be objects associated to a specific "Type" (in this case "Name" = "Type") or elements associated to a specific "Name" in the G_DECL for that "Type", extracted from "G_OBJ".

***Construction of a G_OBJ:*** $\gamma(Type, Sequence)$
This operation produces a new G_OBJ that has "Type" as type. The new G_OBJ is obtained following the G_DECL for that "Type". The objects of the new G_OBJ are in the sequence "Sequence".

**Table 3.** A small set of association rules

| IdRule | Support | Confidence | Body | Head |
|---|---|---|---|---|
| Rule1 | 0.5 | 0.5 | I1 | I6 |
| Rule2 | 0.5 | 0.5 | I7,I9 | I10 |
| Rule3 | 0.5 | 0.5 | I8 | I5,I7 |
| Rule4 | 0.5 | 0.5 | I6,I3 | I4,I12 |

***Extraction from a Sequence:*** $\delta(Sequence, Position)$
This operation extracts the element in position "Position" from the sequence "Sequence".

***Elimination of an Element from a Sequence:*** $\mu(Sequence, Element)$
The result of this operation is the sequence "Sequence" without the first element "Element".

***Selection of a Sequence:*** $\rho(Sequence1, Sequence2)$
This operation is usually applied when the elements of "Sequence1" are sequences themselves. "Sequence2" represents a list of position, i.e. position indexes. When the operation is applied, the result is a sequence of sequences, each containing only the elements, if present, corresponding to the positions given by "Sequence2". If "Sequence2" refers to a single position, then the result is a sequence of elements. When an element of "Sequence1" is not a sequence itself, it is discarded by the operation.

***Append of Sequences:*** $\alpha(Sequence1, Sequence2)$
The result of this operation is a sequence obtained by appending "Sequence2" to "Sequence1".

***Find an Element in a Sequence:*** $\tau(Sequence, Element)$
The result of this operation is the position of the element "Element" in the sequence "Sequence".

***Restriction of a Sequence:*** $\sigma(Condition, Position, Sequence)$
The result of this operation is a sequence composed by the elements in "Sequence" that in position "Position" verify the condition "Condition".
The condition can be a constraint on the value of an element - in this case it is expressed as a comparison with a constant belonging to the element value domain (e.g.: c, c, = c.....) - or the possibility that an element can assume a certain value (e.g.:Value or ¬Value) or a constraint on the number of values which an element can have - in this case the condition is expressed as a comparison operator, followed by the symbol "e", with a constant (e.g.: $\leq_e$e c, $\geq_e$e c, $=_e$ c,...).

With the composition of the elementary operations, conditionals and recursion, it is possible to express the manipulation of the representations and to transform a type of knowledge into another one.
As running example we are going to consider the query shown in the example not taking under consideration the operations that refers to a call of a mining algorithm.

– `take all by weather if windy == false` is equivalent to

$$\gamma(Table, ((\delta(\varepsilon(Table, Attributes, weather), 1), \delta(\varepsilon(Table, AttrNum, weather), 1),$$
$$\sigma(= false, \tau(\delta(\varepsilon(Table, Attributes, weather), 1), windy),$$
$$\delta(\varepsilon(Table, Values, weather), 1)))))$$

Let us see in more detail the combination of the operation.
$\delta(\varepsilon(Table, Attributes, weather), 1)$ returns the element "Attributes" of the

new G_OBJ that is the sequence of the attributes names (A), i.e. (outlook,temperature,humidity,windy,play).
$\delta(\varepsilon(Table, AttrNum, weather), 1)$ returns the element "AttrNum" of the new G_OBJ that is the element representing the number of the attributes in the table weather.
Let B, C and D be $\mathbf{B} \equiv \tau(\delta(A, 1), windy)$, $\mathbf{C} \equiv \varepsilon(Table, Values, weather)$ and $\mathbf{D} \equiv \delta(C, 1)$. By the definitions of the operations, follows that: B is an element representing the position of the attribute named windy into the sequence A; C is a sequence containing every tuple of the table. C is composed by an element, C1, that is a sequence of elements. Every element of C1 is a sequence containing five values: one for each attribute in a row of the table. The number of elements of C1 is equal to the number of rows of the considered table. D is the sequence C1.
$\sigma(= false, 4, D)$ returns the sequence of elements of D that satisfy the condition for the fourth element. Let this sequence be E.
$\gamma(Table, ((A, E)))$ returns the new table containing only the tuples of weather with false for the field windy.

As said, the operation return a new table that we call TS.

- Recalling that AD is a classifier obtained by the tuples in TS, Use AD With weather is equivalent to:

$$\begin{aligned}&\gamma(Table, ((\alpha(\delta(\varepsilon(Table, Attributes, weather), 1), \varepsilon(Tree, Name, AD)),\\ &\delta(\varepsilon(Table, AttrNum, weather), 1) + 1, Labels(\delta(\varepsilon(Table, Values, weather), 1),\\ &Classify(\delta(\varepsilon(Tree, MaxP, AD), 1), null, weather, \delta(\varepsilon(Tree, Element, AD), 1)),\\ &\delta(\varepsilon(Table, AttrNum, weather), 1), \delta(\varepsilon(Tree, Type, AD), 1)))))\end{aligned}$$

Skipping their formal definitions,we can say that the sub functions Labels and Classify are needed to associate to every tuple its classification value. The result of the **Use-With** operation is a new table with an added attribute: the classification one. The resulting table is called IC.

- `Misclassified IC` is equivalent to

$$\begin{aligned}&\gamma(Table, ((\mu(\delta(\varepsilon(Table, Attributes, IC), 1), \delta(\delta(\varepsilon(Table, Attributes, IC), 1),\\ &\delta(\varepsilon(Table, AttrNum, IC), 1))), \delta(\varepsilon(Table, AttrNum, IC), 1) - 1,\\ &Misc(null, \delta(\varepsilon(Table, Values, IC), 1), \delta(\varepsilon(Table, AttrNum, IC), 1)))))\end{aligned}$$

The wrongly classified instances are returned by the **Misc** sub function. As the result of the **Misclassified** operation we obtain a new table in which every instance has been wrongly classifed by the classification tree.

This approach in modelling the semantics of our query language unveils its algebraic nature, and it allows us to prove basic properties of the operators. As an example of algebraic property that can be proven consider the following one:

$$Misc(A1, T) \cup Misc(A2, T) \equiv Misc(Or(A1, A2), T) \cup Misc(And(A1, A2), T)$$

where:

**Misc(Ai,T)** ≡ Misclassified(Use Ai With T)
**Misc(Or(A1,A2),T)** ≡ Misclassified(Use(Exec Or(A1,A2))With T)
**Misc(And(A1,A2),T)** ≡ Misclassified(Use(Exec And(A1,A2))With T)
**A1, A2** ≡ decision trees with "null" type (they are not composition of trees)
**T** ≡ Table
∪ stands for the concatenation of tables

The proof, that can be carried on by induction on the number of wrongly classified instances, is detailed in[2].

## 4 System Architecture

The kernel of the system is KDDML[1], an environment based on XML[18] in which it is possible to combine several types of knowledge extraction operations. The detailed architecture of KDDML is shown in Fig. 4. In KDDML, in fact, both knowledge extraction problems and their results are represented as XML documents. This makes the problem independent from the applications used to obtain the solution. As we see in Fig. 4 the KDDML environment is composed of several components that are:

**XML4J** [19] external component, used to support XML/DOM neutral representation.

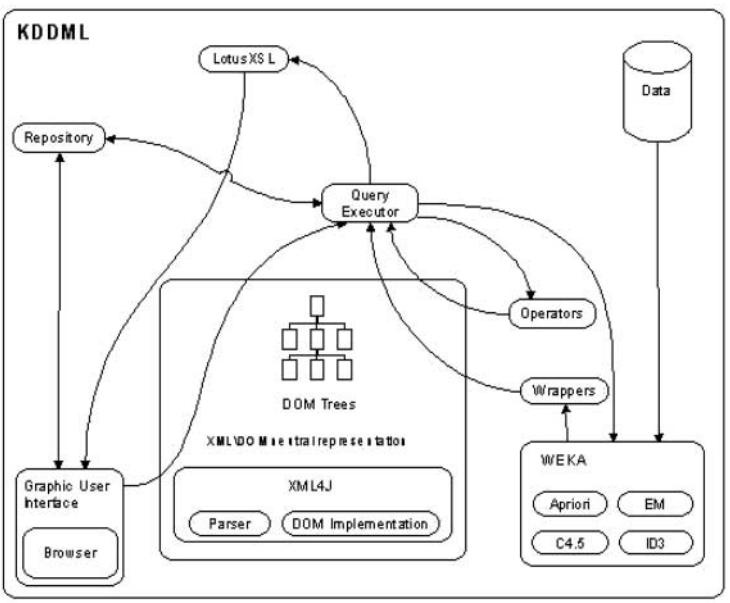



**Fig. 2.** KDDML Architecture

**WEKA** [17] external component, is a Java based package of data mining tools.
**LotusXSL** [10] external component, used to transform XML documents into HTML ones.
**Query Executor** is used to solve knowledge extraction problems expressed by a query in the formalism defined for KDDML.
**Graphical User Interface** is used to allow the user to interact with the environment. It is composed of a component to handle knowledge extraction problems and to retrieve results of a problem, and of a browser for the visualization of the results represented as HTML pages.
**Wrapper** is used to transform the results produced by WEKA into DOM trees and to return the result to the Query Executor.
**Operators** are used to provide the implementation of the operators defined in KDDML language.
**Repository** is used to store queries and their results.

MQL queries are translated into the XML based formalism accepted by KDDML. This is done by the translator component whose main components are:

**compiler** takes an MQL query and returns, as output, a set of KDD_QUERYs: one for each assignment operator (i.e. "=") occurring in the input query. Each KDD_QUERY is then stored in the KDD_QUERY repository;
**MQL query executor** takes the previously generated KDD_QUERY in the correct order. This component uses the KDDML query executor for the actual execution of the query. The result returned by KDDML is a boolean value denoting the success or the failure of the operation.

All the KDDML components are used for the execution of KDD_QUERY (see Fig. 4), except for the user interface, that is designed for the construction of MQL queries.

## 5 Conclusions

The design of the system described in this paper has been prompted by two needs: the need at looking at the KDD process as a query process, and the need of fostering the interoperability between different tools used in the process. The first motivation is shared by many other approaches. A first group of proposals proposes extensions to SQL. In this set there are: *Mine Rule operator*[11], that is capable of expressing many kind of problems concerning the mining of association rules, but it is used to generate rules only, as opposed to querying persistently stored rules. *Mine-SQL*[13] that allows users to express different types of rule queries for discovering association, characteristic, classification and discriminant rules. Mine-SQL also allows to specify the generalized, multiple-level and multiple-attribute association rules as well as specify rule queries over user-defined attributes. *MSQL*[8], is a rule query language that satisfies: the ability to nest SQL (the user can utilise SQL primitives within MSQL); closure (the language provides operations to further manipulate the result of the previous

query); cross-over between data and rules (the language allows primitives which can map generated rules back to the source data, and vice-versa); generation vs querying (the language allows a user to express rule generation vs rule querying using the same expression syntax). *DMQL*[7], allows the specification of several rule generating tasks such as: association, discriminant, classification and characteristic rules. With DMQL you can also specify concept hierarchies and apply them during the mining task. *WEBMINER*[6] extends SQL with primitives that allows the user to provide guidance to the mining engine by specifying the patterns of interest. Another proposal that, differently from the ones reported above, works on object oriented batabases is ODMQL[5] (an extension to OQL). This language allows the user both to apply all the features of DMQL on object-oriented databases and to exploit the inheritance which is typical of this kind of models.

Other proposals are more aimed at embodying knowledge based techniques into a data mining query language. Two examples are Knowledge Access Suite[16], which offers a Pattern Query Language, that allows knowledge based queries, and Rutgers Data Mining Project[12] that aims at creating a second generation of data-mining tools, where users perform data mining activities by specifying high-level information requests in a data-mining query language.

Our method differs from these proposals as we have included in our language different kind of models and we have privileged the expandability of the system. With MQL, in fact, it is simple to add a new model type and, thanks to the general representation, you can manipulate it using the elementary operations. Furthermore MQL is not strictly designed for relational databases, so that you can extract models also from other types of data representation. Following this line, we are currently adding the possibility of querying XML documents from within MQL, exploiting XQUERY[20]. An interesting application area for these extension is information extraction applied to literary texts[3]. In this application, for example, one can extract, via XQUERY, information from annotated texts, and then can perform inductive data mining steps - typically classification and clustering - on the results. Other experiments with the system are in the field of web mining by analysis of logs files of web servers[4].

Also the second motivation is shared by other proposals. PMML (Predictive Model Markup Language)[14] is an XML-based language that uses XML to represent mining models. The structure of the models is described by a DTD (Document Type Definition). PMML includes more model types than MQL-KDDML but, as we said before, it is easy to expand our system to include new models. Given that the model representation of PMML is in the process of becoming a standard, we are planning to adopt it in future releases of our system.

In any case we believe that the algebraic query language approach of MQL is a step forward in the definition of a suitable KDD environment. Several are the necessary extension to obtain this goal. First of all there is the need to extend the language with features for pre-processing data. We are currently approaching this issue trying to mantain the algebraic approach. Secondly it would be helpful

to be able to represent more complex background knowledge then the hierarchies we are currently offering. To this purpose we are considering the possibility of extending the language with the notion of declarative rules.

## References


[1] P. Alcamo, F. Domenichini, and F. Turini. An XML based environment in support of the overall KDD process. In *Proceedings of the Fourth International Conference on Flexible Query Answering Systems*, Advances in Soft Computing, pages 413–424. Springer-Verlag, 2000. 226, 232

[2] M. Baglioni. MQL: una proposta di query language per data mining, 2001. Tesi di Laurea, Dipartimento di Informatica, Univ. di Pisa. 232

[3] M. Baglioni, A. Cappelli, M. N. Catarsi, P. Michelassi, L. Moretti, M. Tavoni, and F. Turini. Knowledge mining and discovery for searching literary texts. In *Proc. of International COnference on Language Resources and Evaluation*, Las Palmas, 2002. 234

[4] D. Bruno. Estensione e sperimentazione di un sistema di knowledge discovery basato su XML, 2001. Tesi di Laurea, Dipartimento di Informatica, Univ. di Pisa. 234

[5] Mohamed G. Elfeky, Amani A. Saad, and Souheir A. Fouad. ODMQL: Object data mining query language. In K. R. Dittrich et al., editor, *Objects and Databases 2000*, volume 1944 of *LNCS*, pages 128–140. Springer-Verlag, 2001. 234

[6] Jiawei Han, Yongjian Fu, Wei Wang, Jenny Chiang, Wan Gong, Krzysztof Koperski, Deyi Li, Yijun Lu, Amynmohamed Rajan, Nebojsa Stefanovic, Betty Xia, and Osmar R. Zaiane. DBMiner: A system for mining knowledge in large relational databases. In *Proc. 1996 Int'l Conf. on Data Mining and Knowledge Discovery (KDD'96)*, pages 250–255, Portland, Oregon, 1996. 234

[7] Jiawei Han, Yongjian Fu, Wei Wang, Krzysztof Koperski, and Osmar Zaiane. DMQL: A data mining query language for relational databases. In *SIGMOD'96 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD'96)*, Montreal, Canada, 1996. 234

[8] Tomasz Imielinshi and Aashu Virmani. Msql: A query language for database mining. In *Data Mining and Knowledge Discovery*, pages 373–408, 1999. 233

[9] T. Imielinski and H. Mannila. A database perspective on knowledge discovery. *Comm. Of The Acm*, 39:58–64, 1996. 225, 226

[10] Lotus XSL. `http://www.alphaworks.ibm.com/formula/xsl`. 233

[11] Rosa Meo, Giuseppe Psaila, and Stefano Ceri. An extension to SQL for mining association rules. In *Data Mining and Knowledge Discovery*, pages 195–224, June 1998. 233

[12] Rutgers Data Mining. http://www.cs.rutgers.edu/ mcgrew/explorer/1.3/mine-over-data.html. 234

[13] Tadeusz Morzy and Maciej Zakrzewicz. Sql-like language for database mining. In *Proceedings of the First East-European Symposium on Advances in Databases and Information Systems (ADBIS'97), St.-Petersburg, September 2-5, 1997. Volume 1: Regular Papers*, pages 311–317. Nevsky Dialect, 1997. 233

[14] Predictive Model Markup Language (PMML). Version 2.0. `http://www.dmg.org`. 234

[15] J. R. Quinlan. C4.5: Programs for machine learning. *Morgan Kaufmann Publ. Inc.*, 1993. 226

[16] Knowledge Access Suite. http://www.datamining.com/ka-suite.htm. 234
[17] Weka: machine learning software in java. Open source software. http://www.cs.waikato.ac.nz/∼ml/weka/. 233
[18] Extensible Markup Language (XML) 1.0 (second edition). W3C Recommendation. Oct. 6, 2000, http://www.w3.org/TR/REC-xml. 232
[19] IBM's Alphaworks XML4J. http://www.alphaworks.ibm.com/formula/xml. 232
[20] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery. 234

# Preprocessing and Mining Web Log Data for Web Personalization

M. Baglioni[1], U. Ferrara[2], A. Romei[1], S. Ruggieri[1], and F. Turini[1]

[1] Dipartimento di Informatica
Università di Pisa
Via F. Buonarroti 2, 56125 Pisa Italy
{baglioni,romei,ruggieri,turini}@di.unipi.it
[2] KSolutions S.p.A
Via Lenin 132/26, 56017 S. Martino Ulmiano (PI) Italy
ferrara@ksolutions.it

**Abstract.** We describe the web usage mining activities of an on-going project, called ClickWorld[1], that aims at extracting models of the navigational behaviour of a web site users. The models are inferred from the access logs of a web server by means of data and web mining techniques. The extracted knowledge is deployed to the purpose of offering a personalized and proactive view of the web services to users. We first describe the preprocessing steps on access logs necessary to clean, select and prepare data for knowledge extraction. Then we show two sets of experiments: the first one tries to predict the sex of a user based on the visited web pages, and the second one tries to predict whether a user might be interested in visiting a section of the site.

**Keywords:** Knowledge discovery, web mining, classification.

## 1 Introduction

According to [10], *Web Mining* is *the use of data mining techniques to automatically discover and extract information from web documents and services.* A common taxonomy of web mining defines three main research lines: *content mining*, *structure mining* and *usage mining*. The distinction between those categories is not a clear cut, and very often approaches use combination of techniques from different categories.

*Content mining* covers data mining techniques to extract models from web object contents including plain text, semi-structured documents (e.g., HTML or XML), structured documents (digital libraries), dynamic documents, multimedia documents. The extracted models are used to classify web objects, to extract keywords for use in information retrieval, to infer structure of semi-structured or unstructured objects.

[1] Progetto di ricerca industriale *ClickWorld – Tecniche innovative di Web Proactive Personalization a fini di Internet marketing, commercio elettronico, editoria on-line*, Codice Progetto: S585-P, Ente finanziatore: MIUR Legge 488/92.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 237–249, 2003.
© Springer-Verlag Berlin Heidelberg 2003

*Structure Mining* aims at finding the underlying topology of the interconnections between web objects. The model built can be used to categorize and to rank web sites, and also to find out similarity between them.

*Usage mining* is the application of data mining techniques to discover usage patterns from web data. Data is usually collected from user's interaction with the web, e.g. web/proxy server logs, user queries, registration data. Usage mining tools [3, 4, 9, 15] discover and predict user behavior, in order to help the designer to improve the web site, to attract visitors, or to give regular users a personalized and adaptive service.

In this paper, we describe the web usage mining activities of an on-going project, called `ClickWorld`, that aims at extracting models of the navigational behavior of users for the purpose of web site personalization [6]. We have collected and preprocessed access logs from a medium-large national web portal, `vivacity.it`, over a period of five months. The portal includes a national area (`www.vivacity.it`) with news, forums, jokes, etc., and more than 30 local areas (e.g., `www.roma.vivacity.it`) with city-specific information, such as local news, restaurant addresses, theatre programming, bus timetable, ecc.

The preprocessing steps include data selection, cleaning and transformation and the identification of users and of *user sessions* [2]. The result of preprocessing is a data mart of web accesses and registration information. Starting from preprocessed data, web mining aims at pattern discovery by adapting methods from statistics, data mining, machine learning and pattern recognition. Among the basic data mining techniques [7], we mention *association rules*, discovering groups of objects that are frequently requested together by users; *clustering*, grouping users with similar browsing patterns, or grouping objects with similar content or access patterns; *classification*, where a profile is built for users belonging to a given class or category; and *sequential patterns*, namely sequences of requests which are common for many users.

In the `ClickWorld` project, several of the mentioned methods are currently being used to extract useful information for proactive personalization of web sites. In this paper, we describe two sets of classification experiments. The first one aims at extracting a classification model able to discriminate the sex of a user based on the set of web pages visited. The second experiment aims at extracting a classification model able to discriminate those users that visit pages regarding e.g. sport or finance from those that typically do not.

## 2 Preprocessing for Web Personalization

We have developed a data mart of web logs specifically to support web personalization analysis. The data mart is populated starting from a web log data warehouse (such as those described in [8, 16]) or, more simply, from raw web/proxy server log files. In this section, we describe a number of preprocessing and coding steps performed for data selection, comprehension, cleaning and transformation. While some of them are general data preparation steps for web usage mining [2, 16], it is worth noting that in many of them a form of domain knowl-

edge must necessarily be included in order to clean, correct and complete the input data according to the web personalization requirements.

### 2.1 User Registration Data

In addition to web access logs, our given input includes personal data on a subset of users, namely those who are registered to the `vivacity.it` website (registration is not mandatory). For a registered user, the system records the following information: sex, city, province, civil status, born date. This information is provided by the user in a web form at the time of registration and, as one could expect, the quality of data is up to the user fairness. As preprocessing steps, improbable data are detected and removed, such as born data in the future or in the remote past. Also, some additional input fields were not imported in the data mart since almost all values were left as the default choice in the web form. In other words, the fields were considered not to be useful in discriminating user choices and preferences.

In order to avoid users to digit their login and password at each visit, the `vivacity.it` web site adopts cookies. If a cookie is provided by the user browser, then authentication is not required. Otherwise, after authentication, a new cookie is sent to the user browser. With this mechanism, it is possible to track any user as long as she deletes the cookies on her system. In addition, if the user is registered, the association login-cookie is available in the input data, and then it is possible to track the user also after she deletes the cookies.

This mechanism allows for detecting non-human users, such as system diagnosis and monitoring programs. By checking the number of cookies assigned to each user, we discovered that the user login 'test009' was assigned more than 24.000 distinct cookies. This is possible only if the user is some program that automatically deletes assigned cookies, e.g. a system diagnosis program.

### 2.2 Web URL

Resources in the World Wide Web are uniformly identified by means of URLs (Uniform Resource Locators). The syntax of an http URL is:

```
'http://' host.domain [':'port] [ abs_path ['?' query]]
```

where

- `host.domain[:port]` is the name of the server site. The TCP/IP `port` is optional (the default port is 80),
- `abs_path` is the absolute path of the requested resource in the server filesystem. We further consider `abs_path` of the form `path '/' filename ['.' extension]`, i.e. consisting of the filesystem path, filename and file extension.
- `query` is an optional collection of parameters, to be passed as an input to a resource that is actually an executable program, e.g. a CGI script.

On the one side, there are a number of normalizations that must be performed on URLs, in order to remove irrelevant syntactic differences (e.g., the host can be in IP format or host format – `131.114.2.91` is the same host as `kdd.di.unipi.it`). On the other side, there are some web server programs that adopt non-standard formats for passing parameters. The `vivacity.it` web server program is one of them. For instance, in the following URL:

```
http://roma.vivacity.it/speciali/EditColonnaSpeciale/1,3478,|DX,00.html
```

the file name `1,3478,|DX,00` contains a code for the local web site (`1` stands for `roma.vivacity.it`), a web page id (`3478`) and its specific parameters (`DX`).

The form above has been designed for efficient machine processing. For instance, the web page id is a key for a database table where the page template is found, while the parameters allow for retrieving the web page content in some other table. Unfortunately, this is a nightmare when mining clickstream of URLs. Syntactic features of URLs are of little help: we need some semantic information, or ontology [5, 13], assigned to URLs.

At the best, we can expect that an *application-level* log is available, i.e. a log of accesses to semantic-relevant objects. An example of application-level log is one recording that the user entered the site from the home page, then visited a sport page with news on a soccer team, and so on. This would require a system module monitoring user steps at a semantic level of granularity. In the `ClickWorld` project such a module is called `ClickObserve`. Unfortunately, however, the module is a deliverable of the project, and it was not available for collecting data at the beginning of the project.

Therefore, we decided to extract both syntactic and semantic information from URLs via a semi-automatic approach. The adopted approach consists in *reverse-engineering* URLs, starting from the web site designer description of the meaning of each URL path, web page id and web page parameters. Using a PERL script, starting from the designer description we extracted from original URLs the following information:

- local web server (i.e., `vivacity.it` or `roma.vivacity.it` etc.), which provides us with some spatial information about user interests;
- a first-level classification of URLs into 24 types, some of which are: home, news, finance, photogalleries, jokes, shopping, forum, pubs;
- a second-level classification of URLs depending on the first-level one, e.g. URLs classified as *shopping* may be further classified as *book shopping* or *pc shopping* and so on;
- a third-level classification of URLs depending on the second-level one, e.g. URLs classified as *book shopping* may be further classified as *programming book shopping* or *narrative book shopping* and so on;
- a parameter information, further detailing the three level classification, e.g. URLs classified as *programming book shopping* may have the ISBN book code as parameter;
- the depth of the classification, i.e. 1 if the URL has only a first-level classification, 2 if the URL has first and second-level classification, and so on.

Of course, the adopted approach was mainly an heuristics one, with the hierarchical ontology designed at posteriori. Also, the designed ontology does not exploit any content-based classification, i.e. the description of an elementary object such as *sport news with id 12345* is its code (i.e., first-level is *news*, second-level is *sport*, parameter information *12345*), with no reference to the content of the news (was the news reporting about any specific player?).

### 2.3 Web Logs

Log files produced by web/proxy servers are text files with a row for each HTTP transaction. A typical row contains the following information:

```
...  213.213.31.41 [15/Apr/2000:04:00:04 +0200]
  "GET http://www.unipi.it/images/h/h_home.gif HTTP/1.1" 200 1267
      MmTaUg00pdA00001fvkwsM4000 http://www.unipi.it MSIE+6.0 ...
```

Here, `213.213.31.41` is the IP address of the client (the computer originating the request), `15/Apr/2000:04:00:04` is the date/time of the transaction, `GET` is the *method* of the transaction (we consider requests concerning `GET` and `POST`), `http://www.unipi.it/images/h/h_home.gif` is the URL requested by the client, `HTTP/1.1` is the HTTP protocol, `200` is the HTTP return code (`200` means OK), `1267` is the size in bytes of the response sent to the client, `MmTaUg00pdA00001fvkwsM4000` is the cookie at the client browser, `http://www.unipi.it` is the referrer, i.e. the URL referring to the requested one, and `MSIE+6.0` is the client environment specification provided by the client browser. The number of HTTP transactions per day of a medium-large web server is in the order of millions of requests. However, we are only interested in requests of `.html` files, which can be assigned a semantic meaning, as described in the last section. The `vivacity.it` dataset records requests for a period of 5 months for a total of more than 7 millions of requests. Peaks are distributed over working days.

HTTP transactions are preprocessed by hash coding URLs, IPs, client environment and cookies, removing web robots requests and system requests (i.e., "pings" from monitoring modules, batch requests and administration requests), loading them into a relational database, and computing the following derived attributes:

- `Last` and `Next` count how long in the past and in the future the same URL is requested. The count is absolute, i.e. it considers the number of requests. `Last` (resp., `Next`) is set to a large constant $\infty$ if there is no previous (resp., further) request.
- `PageDelay` is the distance in milliseconds of a request from the previous request of the same user. `PageDelay` is set to a large constant $\infty$ in case of the very first request of an user.
- `SessionID` is a unique identifier of the user session the request belongs to.

It is worth discussing on the notions of *user* and *user session*.

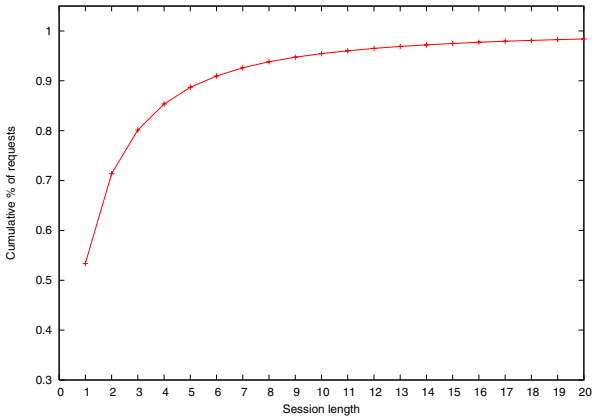



**Fig. 1.** Distribution of requests over session length

We cannot use the login name as an identifier for users. In fact, the login name is known only for registered users. As a first heuristics, we could use cookies as user identifiers. On the one hand, this is an approximation: once a user deletes her cookies, she will appear as a distinct user. On the other hand, web browsers can be set not to accept cookies. The former approximation cannot be refined. For the latter, one could consider the pair (cookie, IP) in case the cookie is not present. Considering only the cookie leads to have a single user ("those who do no accept cookies") with more than 20% of all requests. Nevertheless, the pair (cookie, IP) can only track a single "visit" (what we are going to call a *user session*) of a user and cannot be used to track users along two or more visits (IP can be reassigned, users can use different computers at home and at work, etc.).

Consider now the concept of user session. From a semantical point of view, a user session could be defined as the set of URLs accessed by a user for a particular purpose. We do not try to infer (or guess) the "purpose" in mind of the user, but rather we consider some heuristics [2]. Here, we adopt the *reference length* approach, which is based on the assumption that the amount of time a user spends examining an object is related to the interest of the user for the object contents. On this basis, a model for user sessions is obtained by distinguishing the *navigational* objects (i.e., containing only links interesting to the user) from the *content* objects (i.e., containing the information the user was looking for). The distinction between navigational and content accesses is related to the distance (in time) between a request and the next one. If between two accesses `A` and `B` there is a time delay greater than a given threshold, then `A` can be considered as a *content* URL; otherwise, it is a *navigational* URL.

On this basis, a user session is a sequence of navigational URLs followed by one content URL. Starting from `PageDelay`, we assign a `SessionID` to each URL request by grouping requests whose distance is below a given threshold. Fig-

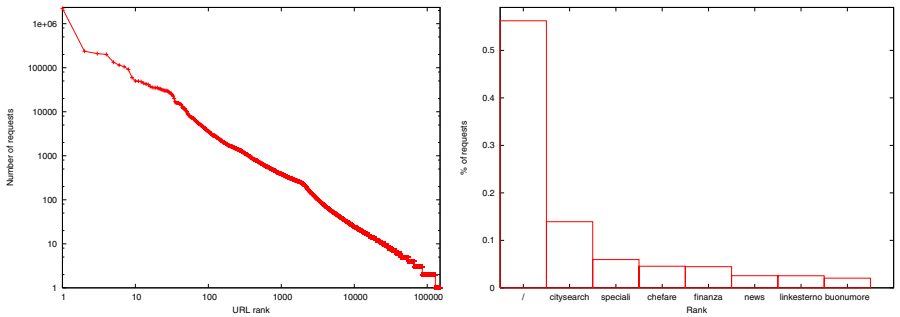



**Fig. 2.** Distribution of requests over URL rank and over first-level semantic rank

ure 2.3 reports the distribution of requests over session length, where a threshold of 30 seconds is considered in computing sessions.

As a final consideration, it is well-known that the distribution of web accesses over URL typically follows the Zipf's law, which states that $f = r^{-\alpha}$ where $f$ is the number of requests to an URL, $r$ is the URL rank (i.e., 1 for the most accessed URL, 2 for the second most accessed one, and so on) and $\alpha$ is a parameter. Figure 2.3 (left) shows on the X axis the URLs rank and on the Y axis the number of requests to that URL. The plot is exponential on both the X and Y axis. The straight line confirms the Zipf's law, since (by passing to logarithms) $log\ f = -\alpha log\ r$. A law similar to Zipf's can be expected at *semantic level*. Figure 2.3 (right) shows the distribution of requests over first-level semantic of accessed URLs.

## 3 Classification Experiments

In the `ClickWorld` project, we are conducting experiments to infer knowledge from access logs in order to provide web site users a proactive and personalized view. In this section, we describe two sets of classification experiments.

The first one tries to predict the sex of a user based on the web pages visited. We aim at extracting a classification model able to discriminate sex based on the navigational behavior of users. The knowledge of the sex allows to target banners, promotions, news, or to give relevance to specific sections of the web site. Similarly, the age of a user could also be relevant to be predicted, but the limited results on sex inference prevented us from exploring age inference.

The second set of experiments tries to predict whether a user might be interested in visiting a section of the web site based on the sections the user has already visited. The knowledge of the sections that could be of interest to the user allows for on-the-fly menu or page reorganization, thus giving a personalized view of the web site.

### 3.1 Predicting Sex from Access Logs

Only a small subset (about 15%) of the users in our dataset has registration information. It is then appealing to be able to build from those users a classification model that predicts sex of non-registered users. In this way, the web site could provide personalization features without forcing registration (for instance, this would attract users who would give up if asked for registration).

**Model Extraction Process.** A classification algorithm takes as input a *training set*, which is a set of *cases*, or *tuples* in the database terminology. Each case specifies values for a collection of *attributes* and for a *class*. The output of the classification algorithm is a model that describes/predicts the class value of a case on the basis of the values of the attributes of the case. The predictive accuracy of the extracted model is evaluated on a *test set*, for which the actual class is known.

In our context, the set of registered users is split into training and test set (67% and 33% respectively).The attributes of a case consist of the site pages or sections visited by the user and the class consists of the user sex. Actually, assuming an attribute for each site page is not feasible (medium-large web site have hundreds of thousands of pages). As a first alternative, we consider the top most accessed pages. As a second alternative, we consider sections, i.e. we aggregate site pages according to their first-level ontology (or at deeper levels). This way, we have 24 attributes and a class. Each attribute value may be binary (i.e., sport pages have been visited or not) or numeric (i.e., sport pages have been visited $n$ times) or discrete (i.e., sport pages have been visited zero, a few or many times).

Also, a case may be seen at two levels of granularities. The first level is to have a case for each user session, i.e. each case considers the sections visited in a single user session. The second level is to consider a case for recent past sessions of a user, i.e. each case considers the sections visited in the recent past by a user. Data for the second level can be obtained for registered users only, since for non-registered ones we cannot link two sessions of a same user (this happens when the user deletes her cookies). Since we would like to apply the model to predict the sex of non-registered users, we restrict to consider the first level of granularity.

To sum up, we have conducted a large number of experiments by varying all the factors above in the definition of the data set. Also, we have used a few classification models, including decision trees [14], instance-based models [1], classification by association[11] and bagging and boosting meta-classifiers [7]. Decision trees built with the C4.5 algorithm performed the best over almost all the experiments.

**Experimental Results.** The term of comparison in evaluating the experimental results is the *majority* classifier, i.e. the one that always predicts the most frequent class (which in our data is *man*). The error of the majority classifier over

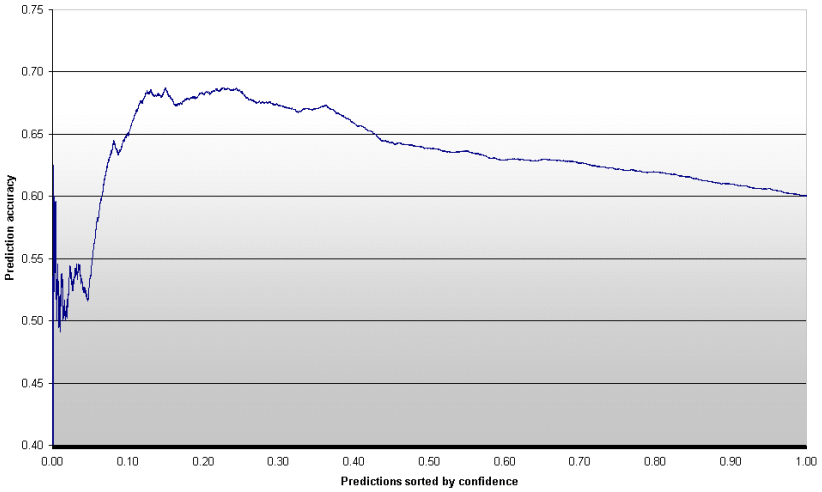



**Fig. 3.** Accuracy of predictions ordered by prediction confidence

the 11614 user sessions of the test set is 45.2%, or, otherwise, its classification accuracy is 54.8%.

As a first experiment, we consider an attribute for each first-level semantics. Attribute values are *yes* or *no* depending whether at least one page belonging to the semantics has been visited in a user session. The C4.5 decision tree classifier yields a misclassification error of 40.7%, while the IBk instance based classifier yields 44.1%. Such numbers cannot be considered satisfactory.

In order to improve accuracy, we tried to provide more informative data, either by adding more attributes (i.e., considering first and second-level semantics) or by considering numeric attribute values (i.e., the number of pages of a given semantics visited in a user session). The latter yielded a higher misclassification rate, while the former slightly reduced it (39.8% for C4.5, and 41.4% for IBk). We also tried to control classifier induction parameters, such as pruning, bagging, boosting and n-fold cross validation, but with no significative improvement.

However, rather than simply looking at the misclassification error, let us consider a kind of lift report, which is shown in Figure 3.1. The plot is obtained as follows. Consider the predictions of the C4.5 decision tree ordered by prediction confidence. This is the X axis. Consider now for the first $n$ predictions, the percentage of these that are correct. This is the Y axis. For any $n$, the majority classifier yields 54.8% in the Y axis! (assuming uniform distribution of cases over the X axis) On the contrary, our classifier yield as much as 65% for the top 10% predictions and as much as 68% for the top 20% predictions. As an application, the classifier yields a lift of 10.2% (resp., 13.2%) over majority classifier when selecting 10% (resp., 20%) of users to whom offer a sex-targeted service, banner or champaign.

To sum up, the small improvement over the majority classifier can be attributed to several reasons. The first one concerns the discriminative power of adopted ontology, i.e. it is too generic for distinguishing pages typically accessed by men from those typically accessed by women. The second one concerns the web site structure and contents. Typical user sessions consists of very few clicks (see Figure 2.3), and, perhaps, contents of the web site do not provide distinguished access patterns based on the sex of users. Finally, we do not exploit *content mining* techniques, i.e. we only consider an *apriori* ontology, while looking into the contents of accessed pages may reveal topics that discriminate between demographic attributes. This is the approach taken in [12], where, however, a lift of 8% over the majority classifier in the top 10% predictions is reported.

### 3.2 Predicting Interests of Users

The `vivacity.it` web site includes sections (or *channels*) about news, sports, shopping, finance, jokes, restaurants and so on. Channels correspond approximatively to the second-level semantics attached to an URL. In this section, we consider the problem of classifying the user sessions that include an access to a channel based on the accesses to the other channels. Such a model can be used to predict potential interest of a new user in a channel she has not visited yet, based on the channels she visited in last or current session. As a consequence, the web site could highlight the channel suggested by the model thus offering a proactive presentation of contents to new users. The same scenario applies when we are going to introduce a *new* channel. Based on the user sessions that autonomously access the new channel, the model could suggest which (old and new) users may be interested in the new channel as well.

**Model Extraction Process.** Since we do not need to restrict to registered users, cases now cover a generic user session and are available in a larger number (more than 215.000 sessions). Fixed a channel as the class, attributes of a case consist of the remaining channels. As in Section 3.1, each attribute value may be binary, numeric or discrete. The class value may be binary (channel accessed or not) or discrete (channel accessed zero, a few or many times). To sum up, we have conducted a large number of experiments by varying all the factors above in the definition of the data set.

It is worth noting that the misclassification error of extracted models cannot be considered as a good quality measure for our purposes. Let $C = \{yes, no\}$ the binary class value. Misclassification error considers those *yes* predictions that are actually *no* and those *no* predictions that are actually *yes*. However, rather than not making errors in *no* predictions, we are more concerned with correct *yes* predictions, i.e. in capturing all interested users and only them! Let $RET$ be the set of cases (from here on, cases in the test set) that have been predicted as *yes*, $REL$ the set of cases that have class *yes*. By borrowing the terminology of information retrieval, we define:

- *recall* as $\#(RET \cap REL)/\#REL$, i.e. the rate of retrieved (by the classification model!) interested users over the total number of interested ones,
- *precision* as $\#(RET \cap REL)/\#RET$, i.e. the rate of retrieved interested users over the total number of retrieved ones.

Recall measures how many users interested in a channel can be predict as such by the classification model. Precision measures how many predictions are correct. As one could expect, maximizing both recall and precision are contrasting objectives.

**Experimental Results.** Since user sessions are short, most of the channels appear in a small fraction of the sessions (typically $< 10\%$). Therefore, the majority classifier would always answer *no*. This means a recall of 0%. In other words, the majority classifier, yet having a good misclassification error, is of little help in capturing potential visitors of a channel.

In Table 1, we report for some of the channels of the web site the recall, the precision and the misclassification error of our best C4.5 decision tree model. Also, we report the misclassification error of the majority classifier. The recall rate is satisfactory, with most of the time nearly the 50%, i.e. in making our predictions we reach 50% of potentially interested users. The precision rate is excellent, i.e. we make very few wrong predictions. Also, the overall misclassification error is typically a half of the misclassification error of the majority classifier.

## 4 Conclusions

We described the web usage mining activities of an on-going project, `ClickWorld`, that aims at extracting models of the navigational behavior of a web site users. Preprocessing is a critical step, where domain knowledge must be necessarily deployed, e.g. in providing an ontology of web page semantics. We have conducted experiments that build classification models for inferring sex and

**Table 1.** Performance of a C4.5 decision tree on the test set

| Channel | Recall | Precision | Misc. error | Maj. misc. error |
|---|---|---|---|---|
| Finance | 44.3% | 98.27% | 3.31% | 5.87% |
| Health | 52.3% | 89.66% | 2.06% | 3.69% |
| Market | 49.1% | 83.34% | 2.20% | 4.05% |
| News | 44.1% | 89.27% | 10.57% | 17.28% |
| Shopping | 31.5% | 91.31% | 5.33% | 7.41% |
| Specials | 60.2% | 92.86% | 11.56% | 26.01% |
| Sport | 50.0% | 91.93% | 1.65% | 2.85% |
| Surveys | 21.9% | 92.66% | 3.92% | 4.91% |
| Theatre | 54.8% | 94.63% | 1.20% | 2.48% |

interests of users based on their navigational behavior. The former models yields a small improvement over random choice, probably due to scarce sex discrimination of access paths in our dataset. Nevertheless, we slightly improve a result reported in the literature. We plan to investigate whether a content mining approach could improve predictive accuracy. The latter models show an excellent precison and a satisfactory recall. All the models do not consider the order of accesses, i.e. user sessions are *sets* and not *sequences*. We are currently considering the issue by investigation sequential patterns, which take into account the order of accesses.

## References


[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991. 244

[2] B. Berendt, B. Mobasher, M. Spiliopoulou, and M. Nakagawa. A framework for the evaluation of session reconstruction heuristics in web usage analysis. *INFORMS Journal of Computing*, 15(2), 2003. 238, 242

[3] B. Berendt and M. Spiliopolou. Analysis of navigation behaviour in web sites integrating multiple information systems. *VLDB Journal*, 9(1):56–75, 2000. 238

[4] R. Cooley, M. Deshpande, J. Srivastava, and P. N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations*, 1(2), January 2000. 238

[5] H. Dai and B. Mobasher. A road map to more effective web personalization: Integrating domain knowledge with web usage mining. In *Proceedings of the International Conference on Internet Computing 2003 (IC03)*, 2003. 240

[6] M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Transactions on Internet Technology (TOIT)*, 3(1):1–27, 2003. 238

[7] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Mateo, CA, 2000. 238, 244

[8] K. P. Joshi, A. Joshi, Y. Yesha, and R. Krishnapuram. Warehousing and mining web logs. In *In Proc. of ACM CIKM Workshop on Web Information and Data Management (WIDM'99)*, pages 63–68. ACM, 1999. 238

[9] KDnuggets. Software for web mining. `http://www.kdnuggets.com/-software/web.html`. 238

[10] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Esplorations*, 2(1):1–15, 2000. 237

[11] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *IEEE International Conference on Data Mining*, pages 369–376, 2001. 244

[12] D. Murray and K. Durrell. Inferring demographic attributes of anonymous internet users. In *Web Usage Analysis and User Profiling Workshop*, volume 1836 of *Lecture Notes in Computer Science*, pages 7–20. Springer, 2000. 246

[13] C. Pohle and M. Spiliopoulou. Building and exploiting ad hoc concept hierarchies for web log analysis. In *Proceedings of DaWaK 2002*, volume 2454 of *LNCS*, pages 83–93, 2002. 240

[14] S. Ruggieri. Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 14:438–444, 2002. 244

[15] M. Spiliopoulou and L. C. Faulstich. WUM: a Web Utilization Miner. In *Proceedings of the EDBT Workshop WebDB98*, volume 1590 of *LNCS*, pages 109–115, 1998. 238

[16] M. Sweiger, M. R. Madsen, J. Langston, and H. Lombard. *Clickstream Data Warehousing*. John Wiley & Sons, 2002. 238

# Multi-relational Structural Bayesian Classifier

Michelangelo Ceci, Annalisa Appice, Donato Malerba, and Vincenzo Colonna

Dipartimento di Informatica, Università degli Studi à di Bari
via Orabona 4, 70126 Bari, Italy
{ceci,appice,malerba}@di.uniba.it
colonna_vin@yahoo.it

**Abstract.** In the traditional naïve Bayes classification method, training data are represented as a single table (or database relation), where each row corresponds to an example and each column to a predictor variable or a target variable. In this paper we propose a multi-relational extension of the naïve Bayes classification method that is characterized by three aspects: first, an integrated approach in the computation of the posterior probabilities for each class; second, the applicability to both discrete and continuous attributes; third, the consideration of knowledge on the data model embedded in the database schema during the generation of classification rules. The proposed method has been implemented in the new system Mr-SBC and tested on three benchmark tasks. Results on predictive accuracy favour our system for the most complex task. Mr-SBC also proved to be an efficient multi-relational data mining system with a tight dose integration to a relational DBMS.

## 1 Introduction

In the usual classification setting, data are generated independently and with an identical distribution from an unknown distribution $P$ on some domain $\mathbf{X}$ and are labelled according to an unknown function $g$. The domain of $g$ is spanned by $m$ independent (or predictor) random variables $X_i$ (both numerical and categorical), that is $\mathbf{X}=\mathrm{X}_1 \times \mathrm{X}_2 \times \ldots \times \mathrm{X}_m$, while the range of $g$ is a finite set $Y=\{\mathrm{C}_1, \mathrm{C}_2, \ldots, \mathrm{C}_L\}$, where each $\mathrm{C}_i$ is a distinct class. An inductive learning algorithm takes a training sample $S=\{(\mathbf{x}, \mathrm{y}) \in \mathbf{X} \times \mathrm{Y} \mid \mathrm{y}=\mathrm{g}(\mathbf{x})\}$ as input, and returns a function $f$ which is hopefully close to $g$ on the domain $\mathbf{X}$.

This problem has received much attention in the literature [15]. A well-known solution is represented by the Naïve Bayesian Classifiers [3], which aim to build a *discriminant function* $d_i$ for each class, such that the value of the real number returned by $d_i(\mathrm{x})$ represents a degree of belief in assigning the observation $x$ to the class $\mathrm{C}_i$. The classification of any $x \in \mathbf{X}$ is the class $\mathrm{C}_k$ maximizing this degree of belief, that is:

$$f(\boldsymbol{x}) = arg\ max_i d_i(x)$$

In the Bayesian framework, $d_i(x)$ estimates the *posterior probability* $P(C_i|x)$ that the observation $x$ is of class $\mathrm{C}_i$. This estimation is based on the training set. By applying the Bayes theorem, $d_i(x)$ can be reformulated as follows:

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 250–261, 2003.
© Springer-Verlag Berlin Heidelberg 2003

$$d_i(x) = P(C_i|x) = \frac{P(C_i)P(x|C_i)}{P(x)}$$

The term $P(x|C_i)$ is in turn estimated by means of the *naïve Bayes assumption.*

$P(x|C_i) = P(x_1, x_2, \ldots, x_m|C_i) = P(x_i|C_i) \times P(x_2|C_i) \times \ldots \times P(x_m|C_i)$

This assumption is clearly false if the predictor variables are statistically dependent. However, even in this case, the naïve Bayesian classifier can give good results [3].

Many inductive learning algorithms assume that the training set can be represented as a single table, where each row corresponds to an example and each column to a predictor variable or to the *target* variable Y. This assumption, also known as *single-table assumption* [20], seems quite restrictive in some data mining applications, where data are stored in a database and are organized into several tables for reasons of efficient storage and access. In this context, both predictor variables and the target variable are represented as attributes of distinct tables (relations).

Although in principle it is possible to consider a single relation reconstructed by performing a relational join operation on the tables, this approach is fraught with many difficulties in practice [2][10]. Consequently, the (multi-)relational data mining approach to the classification problem has been receiving considerable attention in the literature [1][9] [13][18][8].

In this paper we present a new approach to the problem of learning classifiers from relational data. It is based on the induction of a set of first-order classification rules in the context of naive Bayesian classification.

Studies on first-order naïve Bayes classifiers have already been reported in the literature. In particular, Pompe and Kononenko [18] proposed a method based on a two-step process. The first step uses the ILP-R system [17] to learn a hypothesis in the form of a set of first-order rules and then, in the second step, the rules are probabilistically analyzed. During the classification phase, the conditional probability distributions of individual rules are combined naïvely according to the naïve Bayesian formula.

Flach and Lachiche proposed a similar two-step method, however, unlike the previous one, there is no learning of first-order rules in the first step. Alternatively, a set of patterns (first-order conditions) is generated that are used afterwards as attributes in a classical attribute-value naive Bayesian classifier [8]. 1BC, the system implementing this method, views individuals as structured objects and distinguishes between *structural* predicates referring to parts of individuals (e.g. atoms within molecules), and *properties* applying to the individual or one or several of its parts (e.g. a bond between two atoms). An *elementary first-order feature* consists of zero or more structural predicates and one property.

An important aspect of the two (*eager*) approaches is that they keep the phases of first-order rules/conditions generation and of probability estimation separate. In particular, Pompe and Kononenko use ILP-R to induce first-order rules [17] while 1BC uses TERTIUS [7] to generate first order features. Then, the probabilities are computed for each first-order rule or feature. In the classification phase, the two approaches are similar to a multiple classifier because

they combine the results of two algorithms. However, most first-order features or rules share some literals and this approach takes into account the related probabilities more than once. To overcome this problem it is necessary to rely on an integrated approach, so that the computation of probabilities on shared literals can be separated from the computation of probabilities on the remaining literals.

Systems implementing one of the two above approaches work on a set of main-memory Prolog facts. In real-world applications, where facts correspond to tuples stored on relational databases, some pre-processing is required in order to transform tuples into facts. However, this has some disadvantages. First, the set of possible joins between relations in the database constrain the hypothesis space $H$, but for practical reasons after pre-processing only part of the original space $H$ will be represented in the transformed space $H$', so that the best hypothesis in $H$ may not be part of $H$'. Conversely, much of the pre-processing, which is often expensive in terms of computation and storage, may be unnecessary, since that part of the hypothesis space may never be explored, perhaps because of early pruning. Second, in applications where data can frequently change, pre-processing has to be frequently repeated. Finally, database schemas provide the learning system, free of charge, with useful knowledge of a data model that can help to guide the search process. This is an alternative to asking the users to specify a language bias, such as in 1BC.

In this paper the system Mr-SBC (Multi-Relational Structural Bayesian Classifier) is presented. It implements a new learning algorithm based on an integrated approach of first-order classification rules with naive Bayesian classification, in order to separate the computation of probabilities of shared literals from the computation of probabilities for the remaining literals. Moreover, Mr-SBC is tightly integrated with a relational database, and handles categorical as well as numerical data through a discretization method.

The paper is organized as follows. In the next section the problem is introduced and defined. The induction of first-order classification rules is presented in Section 3, the discretization method is explained in Section 4 and the classification model is illustrated in Section 5. Finally, experimental results are reported in Section 6 and some conclusions are drawn.

## 2 Problem Definition

In traditional classification systems that operate on a single relational table, an observation (or individual) is represented as a tuple of the relational table. Conversely, in Mr-SBC, which induces first-order classifiers from data stored in a set $S$={$\mathrm{T}_0$,$\mathrm{T}_1$,...,$\mathrm{T}_h$} of tables of a relational database, an individual is a tuple $t$ of a *target* relation $T$ joined with all the tuples in $S$ which are related to $t$ following a foreign key path. Formally, a foreign key path is defined as follows:

**Definition 1.** *A foreign key path* is an ordered sequence of tables $\vartheta = (T_{i_1}, T_{i_2}, \ldots, T_{i_s})$, where

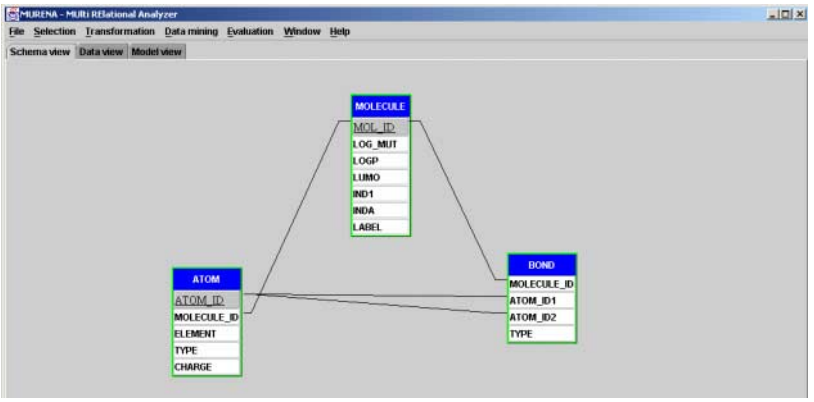



**Fig. 1.** An example of a relational representation of training data of the Mutagenesis database

- $\forall$j=1, ..., $s$, $T_{i_j} \in S$
- $\forall$j=1, ..., $s$-1, $T_{i_{j+1}}$ has a foreign key to the table $T_{i_j}$.

In Figure 1 an example of foreign key paths is reported. In this case, $S$={MOLECULE, ATOM, BOND} and the foreign keys are: A_M_FK, B_M_FK, A_A_FK1, A_A_FK1. If the target relation $T$ is MOLECULE then the possible foreign key paths are:

(MOLECULE)
(MOLECULE,ATOM)
(MOLECULE,BOND)
(MOLECULE, ATOM, BOND)
(MOLECULE, ATOM, BOND)

The last two are equal because the bond table has two foreign keys referencing the table atom.

A formal definition of the learning problem solved by MR-SBC is the following problem:

*Given:*

- A *training set* represented by means of $h$ relational tables $S$={$\mathrm{T}_0$,$\mathrm{T}_1$,...,$\mathrm{T}_h$} of a relational database $D$.
- A set of *primary key constraints* on tables in $S$.
- A set of *foreign key constraints* on tables in $S$.
- A *target relation* $T(\mathrm{x}_1,\ldots,\mathrm{x}_n) \in S$
- a *target* discrete *attribute* $y$ in $T$, different from the primary key of $T$.

*Find:*

A naive Bayesian classifier which predicts the value of $y$ for some individual represented as a tuple in $T$ (with possibly UNKNOWN value for $y$) and related tuples in S according to foreign key paths.

## 3 Generation of First-Order Rules

Let $R$' be a set of first-order classification rules for the classes $\{C_1, C_2, \ldots, C_L\}$, and $I$ an individual to be classified and defined as above. The individual can be logically represented as a set of ground facts, the only exception being the fact associated to the target relation $T$, where the argument corresponding to the target attribute $y$ is a variable $Y$. A rule $R_j \in R$' *covers* $I$, if a substitution $\theta$ exists, such that $R_j\theta \subseteq I\theta$. The application of the substitution to $I$ is required to ground the only variable $Y$ in $I$ to the same constant as that reported in $R_j$ for the target attribute. Let $R$ be the subset of rules in $R$' that cover $I$, that is $R$=$\{R_j \in R' \mid R_j \text{ covers } I\}$. The first-order naïve Bayes classifier for the individual $I$, $f(I)$, is defined as follows:

$f(I) = arg\ max_i P(C_i|R) = arg\ max_i \frac{P(C_i)P(R|C_i)}{P(R)}$

The value $P(C_i)$ is the prior probability of the class $C_i$. Since $P(R)$ is independent of the class $C_i$, it does not affect $f(I)$, that is,

$$f(I) = arg\ max_i \quad P(C_i)P(R|C_i) \tag{1}$$

The computation of $P(R|C_i)$ depends on the structure of $R$. Therefore, it is important to clarify how first-order rules are built in order to associate them with a probability measure. As already pointed out, Pompe and Kononenko use the first-order learning system ILP-R to induce the set of rules $R'$. This approach is very expensive and does not take into account the bias automatically determined by the constraints in the database. On the other hand, Flach and Lachiche use Tertius to determine the structure of *first-order features* on the basis of the structure of the individuals. The system Tertius deals with learning first-order logic rules from data lacking an explicit classification predicate. Consequently, the learned rules are not restricted to predicate definitions as in supervised inductive logic programming. Our solution is similar to that proposed by Flach since the structure of classification rules is determined on the basis of the structure of the individuals. The main difference is that the classification predicate is considered during the generation of the rules.

All predicates in classification rules generated by Mr-SBC are binary and can be of two different types.

**Definition 2.** A binary predicate $p$ is a *structural* predicate associated to a table $T_i \in S$ if a foreign key $FK$ in $T_i$ exists that references a table $T_{i1} \in S$. The first argument of $p$ represents the primary key of $T_{i1}$ and the second argument represents the primary key of $T_i$.

**Definition 3.** A binary predicate $p$ is a *property* predicate associated to a table $T_i \in S$, if the first argument of $p$ represents the primary key of $T_i$ and the second argument represents another attribute in $T_i$ which is neither the primary key of $T_i$ nor a foreign key in $T_i$.

**Definition 4.** A first order classification rule associated to the *foreign key path* $\vartheta$ is a clause in the form:
$p_0(A_1,y)$:- $p_1(A_1,A_2)$, $p_2(A_2,A_3)$, ..., $p_{s-1}(A_{s-1},A_s)$, $p_s(A_s,c)$.
where

1. $p_0$ is a property predicate associated to the target table $T$ and to the target attribute $y$.
2. $\vartheta = (T_{i_1}, T_{i_2}, \ldots, T_{i_s})$ is a *foreign key path* such that for each $k$=1, ..., $s$-1: $p_k$ is a structural predicate associated to the table $T_{i_k}$
3. $p_s$ is a property predicate associated to the table $T_{i_s}$.

An example of a first-order rule is the following:
*molecule_Label(A, active) :- molecule_Atom(A,B), atom_Type(B,'[22..27]').*

Mr-SBC searches all possible classification rules by means of a breadth-first strategy and iterates over some refining steps. A refining step is biased by the possible foreign key paths and consists of the addition of a new literal, the unification of two variables and, in the case of a property predicate, the instantiation of a variable.

The search strategy is biased by the structure of the database because each refining step is made only if the generated first-order classification rule can be associated to a foreign key path. However, the number of refinement steps is upper bounded by a user-defined constant MAX_LEN_PATH.

## 4 Discretization

In Mr-SBC continuous attributes are handled through supervised discretization. Supervised discretization methods utilize the information on the class labels of individuals to partition a numerical interval into bins. The proposed algorithm sorts the observed values of a continuous feature and attempts to greedily divide the domain of the continuous variable into bins, such that each bin contains only instances of one class. Since such a scheme could possibly lead to one bin for each observed real value, the algorithm is constrained to merge bins in a second step. Merging of two contiguous bins is performed when the increase of entropy is lower than a user-defined threshold (MAX_GAIN). This method is a variant of the one-step method 1RD by Holte [12] for the induction of one-level decision trees, that proved to work well with the Naïve Bayes Classifier [4]. It is also different from the one-step method by Fayyad and Irani [5] that recursively splits the initial interval according to the class information entropy measure until a stopping criterion based on the Minimum Description Length (MDL) principle is verified.

## 5 The Computation of Probabilities

According to the naïve Bayes assumption, the attributes are considered independent. However, this assumption is clearly false for the attributes that are

primary keys or foreign keys. This means that the computation of $P(R|C_i)$ in equation (1) depends on the structures of rules in $R$.

For instance, if $R_1$ and $R_2$ are two rules of class $C_i$, that share the same structure and differ only for the property predicates in their bodies

$R_1$: $\beta_{1,0} : -\beta_{1,1}, ..., \beta_{1,K_1-1}, \beta_{1,K_1}$

$R_2$: $\beta_{2,0} : -\beta_{2,1}, ..., \beta_{2,K_2-1}, \beta_{2,K_2}$

where

$$K_1 = K_2 \text{ and } \beta_{1,1} = \beta_{2,1}, \beta_{1,2} = \beta_{2,2}, ..., \beta_{1,K_1-1} = \beta_{2,K_2-1}$$

then $P(\{R_1, R_2\}|C_i) = P(\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1}) \cap \beta_{1,K_1} \cap \beta_{2,K_2}|C_i)=$

$$P(\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1})|C_i) \cdot P(\beta_{1,K_1} \cap \beta_{2,K_2}|\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1}) \cap C_i)$$

The first term takes into account the structure common to both rules, while the second term refers to the conditional probability of satisfying the property predicates in the rules given the common structure.

The latter probability can be factorized under the naïve Bayes assumption, that is:

$$P(\beta_{1,K_1} \cap \beta_{2,K_2}|\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1}) \cap C_i) =$$

$$P(\beta_{1,K_1}|\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1}) \cap C_i) \cdot P(\beta_{2,K_2}|\beta_{1,0} \cap (\beta_{1,1}, ..., \beta_{1,K_1-1}) \cap C_i)$$

According to this approach the conditional probability of the structure is computed only once. This approach differs from that proposed in the works of Pompe and Kononenko [18] and Flach [8] where the factorization would multiply the structure probability twice.

By generalizing to a set of classification rules we have:

$$P(C_i)P(R|C_i) = P(C_i)P(structure)\prod_j P(R_j|structure) \tag{2}$$

where the term *structure* takes into account the class $C_i$ and the structures of the rules in $R$.

If the classification rule $R_j \in R$ is in the form $\beta_{j,0} : -\beta_{j,1}, ..., \beta_{j,K_j-1}, \beta_{j,K_j}$ where $\beta_{j,0}$and $\beta_{j,K_j}$are property predicates and $\beta_{j,1}, \beta_{j,2}, ..., \beta_{j,K_j-1}$are structural predicates, then:

$$P(R_j|structure) = P(\beta_{j,K_j}|\beta_{j,0}, \beta_{j,1}, ..., \beta_{j,K_j-1}) = P(\beta_{j,K_j}|C_i, \beta_{j,1}, ..., \beta_{j,K_j-1})$$

where $C_i$ is the value of the target attribute in the head of the clause ($\beta_{j,0}$). To compute this probability, we use the Laplace estimation:

$$P(\beta_{j,K_j}|C_i, \beta_{j,1}, ..., \beta_{i,K_j-1}) = \frac{\#(\beta_{j,K_j}, C_i, \beta_{j,1}, ..., \beta_{j,K_j-1}) + 1}{\#(C_i, \beta_{j,1}, ..., \beta_{j,K_j-1}) + F}$$

where $F$ is the number of possible values of the attribute to which the $\beta_{j,K_j}$ property predicate is associated. Laplace's estimate is used in order

to avoid null probabilities in the equation (2). In practice, the value at the nominator is the number of individuals which satisfy that conjunction $\beta_{j,K_j}, C_i, \beta_{j,1}, ..., \beta_{j,K_j-1}$, in other words, the number of individuals covered by the rule $\beta_{j,0} : -\beta_{j,1}, ..., \beta_{j,K_j-1}, \beta_{j,K_j}$. It is determined by a "select count (*)" SQL instruction. The value of the denominator is the number of individuals covered by the rule $\beta_{j,0} : -\beta_{j,1}, ..., \beta_{j,K_j-1}$.

The term *P(structure)* in the equation (2) is computed as follows:
Let B=$\{(\beta_{j,1},\beta_{j,2},\ldots,\beta_{j,t})|\ j{=}1..s$ and t=1, ...,$K_j - 1\}$ the set of all distinct sequences of structural predicates in the rules of $R$. Then

$$P(structure) = \prod_{seq \in \mathrm{B}} P(seq) \tag{3}$$

To compute *P(seq)* it is necessary to introduce the definition of the probability JP that a join query is satisfied [11]. Let $\vartheta=(T_{i_1}, T_{i_2},\ldots, T_{i_s})$ be a *Foreign Key Path*, then:

$$\mathrm{JP}(\vartheta) = \mathrm{JP}(\mathrm{T}_{i_1}, ..., \mathrm{T}_{i_s}) = \frac{|\bowtie(\mathrm{T}_{i_1} \times ... \times \mathrm{T}_{i_s})|}{|\ \mathrm{T}_{i_1}| \times ... \times |\mathrm{T}_{i_s}|}$$

where $\bowtie(\mathrm{T}_{i_1} \times ... \times \mathrm{T}_{i_s})$ is the result of the join between the tables $\mathrm{T}_{i_1}, ..., \mathrm{T}_{i_s}$.

We must remember that each sequence *seq* is associated to a foreign key path $\vartheta$. If $seq=(\beta_{j,1},\beta_{j,2}, \ldots,\beta_{j,t})$ there are two possibilities: either a prefix of *seq* is in $B$ or not. By denoting as $\mathrm{T}_{j_h}$ the table related to $\beta_{j,h}$, $h$=1,..., $t$, the probability $P(seq)$ can be recursively defined as follows:

$$P(seq) = \begin{cases} \mathrm{JP}(\mathrm{T}_{j_1}, ..., \mathrm{T}_{j_t}) \text{ if } seq \text{ has no prefix in B} \\ \frac{\mathrm{JP}(\mathrm{T}_{j_1},...,\mathrm{T}_{j_t})}{P(seq')} \quad \text{if } seq' \text{ is the longest prefix of } seq \text{ in B} \end{cases}$$

This formulation is necessary in order to compute the formula (3) considering both dependent and independent events.

Since *P(structure)* takes into account the class, *P(seq)* is computed separately for each class.

## 6 Experimental Results

MR-SBC has been implemented as a module of the system MURENA and has been empirically evaluated on the Mutagenesis datasets. Mutagenesis datasets have been taken from the MLNET repository (`http://www.mlnet.org/`). These datasets are relative to the problem of identifying the mutagenic compounds [16] and have been extensively used to test both inductive logic programming (ILP) systems and (multi-)relational mining systems. We considered, analogously to related experiments in the literature, the "regression friendly" dataset of 188 elements.

A recent study on this database [19] recognizes five levels of background knowledge for mutagenesis which can provide more detailed descriptions of the

**Table 1.** Background knowledge for Mutagenesis database

| **Background** | **Description** |
|---|---|
| $BK_0$ | Consists of those data obtained with the molecular modelling package QUANTA. For each compound it obtains the atoms, bonds, bond types, atom types, and partial charges on atoms. |
| $BK_1$ | Consists of Definitions in $BK_0$ plus indicators *ind1*, and *inda* in molecule table. |
| $BK_2$ | Variables (attributes) *logp*, and *lumo* are added to definitions in $BK_1$. |

examples. In this study we used only the first three levels of background knowledge in order to compare the performance of Mr-SBC with other methods for which experimental results are available in the literature. Table 1 shows the first three sets of background knowledge used in our experiments, where $BK_i \subseteq BK_{i+1}$ for i=0, . . . , 2. The greater the BK, the more complex the learning problem.

The dataset is analyzed by means of a 10-fold cross-validation, that is, the target table is firstly divided into ten blocks of near-equal size and distribution of class values, and then, for every block, a subset of tuples in $S$ related to the tuples in the target table block are extracted. In this way, ten databases are created. Mr-SBC is trained on nine databases and tested on the hold-out database. Mr-SBC has been executed with the following parameters: MAX_LEN_PATH=4 and MAX_GAIN= 0.5.

Experimental results on predictive accuracy are reported in Table 2 for increasing complexity of the models. A comparison to other results reported in the literature is also made. Mr-SBC has the best performance for the most complex task ($BK_2$) with an accuracy of almost 90%, while it ranks third for the simplest task. It is interesting to note that the predictive accuracy increases with the complexity of the background knowledge.

**Table 2.** Accuracy comparison on the set of 188 regression friendly elements of Mutagenesis. Results for Progol2, Foil, Tilde are taken from [1]. Results for Progol_1 are taken from [19]. The results for 1BC are taken from [6]. Results for MRDTL are taken from [14]. The values are the results of 10-fold cross-validation

| *System* | textitAccuracy(%) | | |
|---|---|---|---|
| | $BK_0$ | $BK_1$ | $BK_2$ |
| Progol_1 | 79 | 86 | 86 |
| Progol_2 | 76 | 81 | 86 |
| Foil | 61 | 61 | 83 |
| Tilde | 75 | 79 | 85 |
| MRDTL | 67 | **87** | 88 |
| 1BC | **80.3** | — | 87.2 |
| Mr-SBC | 76.5 | 81 | **89.9** |

**Table 3.** Time comparison of the set of 188 regression friendly elements of Mutagenesis. Results for Progol2, Foil, Tilde are taken from [1]. Results for Progol_1 are taken from [19]. Results for MRDTL are taken from [14]. The results of MR-SBC have been taken on a PIII WIN2k platform

| *System* | *Time (Secs)* | | |
|---|---|---|---|
| | $BK_0$ | $BK_1$ | $BK_2$ |
| Progol_1 | 8695 | 4627 | 4974 |
| Progol_2 | 117000 | 64000 | 42000 |
| Foil | 4950 | 9138 | 0.5 |
| Tilde | 41 | 170 | 142 |
| MRDTL | 0.85 | 170 | 142 |
| 1BC | – | – | – |
| MR-SBC | 36 | 42 | 48 |

**Table 4.** Number of induced rules for the Mutagenesis database

| *FOLD* | *Number of rules* | | |
|---|---|---|---|
| | $BK_0$ | $BK_1$ | $BK_2$ |
| 1 | 57 | 61 | 65 |
| 2 | 56 | 60 | 67 |
| 3 | 56 | 60 | 64 |
| 4 | 56 | 60 | 64 |
| 5 | 57 | 61 | 65 |
| 6 | 53 | 57 | 61 |
| 7 | 57 | 61 | 65 |
| 8 | 57 | 61 | 65 |
| 9 | 54 | 58 | 68 |
| 10 | 56 | 60 | 64 |
| Av. Number | 55,9 | 59,9 | 64,8 |

As regards execution time (see Table 3). The time required by Mr-SBC increases with the complexity of the background knowledge. Mr-SBC is generally considerably faster than competing systems, such as Progol, Foil, Tilde and 1BC, that do not operate on data stored in a database. Moreover, except for the task $BK_0$, Mr-SBC performs better that MRDTL which works on a database. It is noteworthy that the trade-off between accuracy and complexity is in favour of Mr-SBC.

In Table 4 the number of extracted rules for each fold is shown. The number of rules is quite high, although some of them are either redundant or cover very few individuals. Therefore, some additional stopping criteria are required to avoid the generation of these rules and to reduce further the cost complexity of the algorithm.

## 7 Conclusion

In the paper, a multi-relational extension of the naïve Bayes classification method is described. It presents several differences with respect to related works. First, it is based on an integrated approach, so that the contribution of literals shared by several rules to the posterior probability is computed only once. Second, it works both on discrete and continuous attributes. Third, the generation of rules is based on the knowledge of a data model embedded in the database schema. The proposed method has been implemented in the new system Mr-SBC and tested on four benchmark tasks. Results on predictive accuracy are in favour of our system for the most complex tasks. Mr-SBC also proved to be a very efficient multi-relational data mining system with a tight integration to a relational DBMS.

As future work, we plan to extend the comparison of Mr-SBC to other multi-relational data mining systems on a larger set of benchmark datasets. Moreover, we intend to frame the proposed method in a transduction inference setting where both labelled and unlabelled data are available for training.

## Acknowledgements

This work has been supported by the annual Scientific Research Project "Scoperta di conoscenza in basi di dati: metodi e tecniche efficienti e robuste per dati complessi" Year 2002 funded by the University of Bari.

## References

[1] H. Blockeel. *Top-down induction of first order logical decision trees*. PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, 1998. 251, 258, 259

[2] L. De Raedt. Attribute-value learning versus inductive logic programming: the missing links. In *Proceedings of the 8 th International Conference on Inductive Logic Programming*, pages 128–137. Springer-Verlag, 1998. 251

[3] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997. 250, 251

[4] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, 1995. 255

[5] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *In Proc. Of the 13th International Joint Conference on Artificial Intelligence*, pages 1022–1027, 1994. 255

[6] P. Flach and N. Lachiche. First-order bayesian classification with 1BC submitted http://hydria.u-strasbg.fr/ lachiche/1bc.ps.gz, 2000. 258

[7] P. A. Flach and N. Lachiche. Confirmation-guided discovery of first-order rules with tertius. *Machine Learning*, 42(1/2):61–95, 2000. 251

[8] P. A. Flach and N. Lachiche. Decomposing probability distributions on structured individuals. In P. Brito, J. Costa, and Donato Malerba, editors, *Proceedings of the ECML2000 workshop on Dealing with Structured Data in Machine Learning and Statistics*, pages 33–43, Barcelona, Spain, 2000. 251, 256

[9] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In Morgan Kaufman, editor, *In Proceedings of the 6 th International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999. 251

[10] L. Getoor. Multi-relational data mining using probabilistic relational models: research summary. In A. J. Knobbe and D. M. G. van der Wallen, editors, *In Proceedings of the First Workshop in Multi-relational Data Mining*, pages 1300–1309, 2001. 251

[11] L. Getoor, D. Koller, and B. Taskar. Statistical models for relational data. In *In Proceedings of the KDD-2002 Workshop on Multi-Relational Data Mining*, pages 36–55, Edmonton, CA, 2002. 257

[12] R. C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, (11):63–90, 1993. 255

[13] M. Krogel and S. Wrobel. Transformation-based learning using multirelational aggregation. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the 11 th International Conference on Inductive Logic Programming LNAI*, volume 2157. Springer-Verlag, 2001. 251

[14] H. A. Leiva. *MRDTL: A multi-relational decision tree learning algorithm*. PhD thesis, Master thesis, University of Iowa, 2002. 258, 259

[15] T. Mitchell. *Machine Learning*. McGraw Hill, 1997. 250

[16] S. H. Muggleton, M. Bain, J. Hayes-Michie, and D. Michie. An experimental comparison of human and machine learning formalisms. In *In Proc. Sixth International Workshop on Machine Learning*, pages 113–118, San Mateo, CA, 1989. Morgan Kaufmann. 257

[17] U. Pompe and I. Kononenko. Linear space induction in first order logic with relief. In R. Viertl. & G. Della Riccia R. Kruse, editor, *CISM Lecture Notes*. Udine, Italy, 1994. 251

[18] U. Pompe and I. Kononenko. Naive bayesian classifier within ILP-R. In L. De Raedt, editor, *In Proc. of the 5th Int. Workshop on Inductive Logic Programming*, pages 417–436, Dept. of Computer Science, Katholieke Universiteit Leuven, 1995. 251, 256

[19] A. Srinivasan and R. D. Kingand S. Muggleton. The role of background knowledge: using a problem from chemistry to examine the performance of an ilp program, 1999. 257, 258, 259

[20] S. Wrobel. *Relational Data Mining S. Džeroski and N. Lavrac*, chapter Inductive logic programming for knowledge discovery in databases, pages 74–101. Springer, 2001. 251

# Specifying and Verifying Systems of Communicating Agents in a Temporal Action Logic

Laura Giordano[1], Alberto Martelli[2], and Camilla Schwind[3]

[1] Dipartimento di Informatica, Università del Piemonte Orientale, Alessandria
[2] Dipartimento di Informatica, Università di Torino, Torino
[3] MAP, CNRS, Marseille, France

**Abstract.** In this paper we develop a logical framework for specifying and verifying systems of communicating agents. The framework is based on a Dynamic Linear Time Temporal Logic (DLTL). It provides a simple formalization of the communicative actions in terms of their effects and preconditions and the specification of an interaction protocol by means of temporal constraints. We adopt a social approach to agent communication (as proposed by Singh): communication can be described in terms of changes in the social relations between participants, and protocols in terms of creation, manipulation and satisfaction of commitments among agents. The description of the interaction protocol and of communicative actions is given in a temporal action theory, and agent programs, when known, can be specified as complex actions (regular programs in DLTL). The paper addresses several kinds of verification problems (including the problem of compliance of agents to the protocol), which can be formalized either as validity or as satisfiability problems in the temporal logic and can be solved by model checking techniques.

## 1 Introduction

In this paper we develop a logical framework for specifying and verifying systems of communicating agents. The framework relies on the theory for reasoning about action developed in [8] which is based on Dynamic Linear Time Temporal Logic (DLTL). It allows a simple formalization of the communicative actions in terms of their effect and preconditions as well as the specification of an interaction protocol to constrain the behaviours of autonomous agents.

The paper adopts a social approach to agent communication [17, 11]. This approach can be contrasted with the mentalistic approach, which describes the semantics of communication on the base of mental attitudes of the communicating agents, namely, their beliefs, desires and intentions [4] which are represented by modal operators[1]. The mental approach is not well suited for the verification of an "open" multiagent system, where the history of communications is

[1] A closely related approach was proposed by Labrou and Finin [15] to define a semantics for KQML.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 262–274, 2003.
© Springer-Verlag Berlin Heidelberg 2003

observable, but the internal states of the single agents may not be observable. In contrast, in the social approach communicative actions affect the "social state" of the system, rather than the internal states of the agents. The social state records the social facts, like the permissions and the commitments of the agents, which are created and modified in the interactions among them. The dynamics of the system emerges from the interactions of the agents, which must respect these permissions and the commitments (if they are compliant with the protocol). The social approach provides a high level specification of the protocol, and does not require the rigid specification of all the allowed action sequences by means of finite state diagrams, as in a behavioral approach.

Dynamic and temporal logics have recently gained renewed attention in the areas of reasoning about actions and planning [1, 10, 8, 3], fertilizing them with the well known results and techniques developed for such logics. In particular, linear time temporal logic has become a well established tool for specifying the behaviour of distributed systems, for which a rich theory has been developed and the verification task can be automated. In this paper we present a theory for reasoning about communicative actions in a multiagent system which is based on the Product Version of Dynamic Linear Time Temporal Logic (denoted $DLTL^{\otimes}$) [13], an extension of LTL (the propositional linear time temporal logic). It allows to describe the behaviour of a network of sequential agents which coordinate their activities by performing common actions together. $DLTL^{\otimes}$ extends LTL by strengthening the *until* operator by indexing it with the regular programs of dynamic logic. It is, essentially, a dynamic logic equipped with a linear time semantics. It allows the formulas of the logic to be decorated with the names of sequential agents, and it provides a simple way of constraining the (possibly infinite) behaviours of the agents by making use of regular programs. As we will see, regular programs are well suited to model both the agent behaviours and the protocol.

In [13] $DLTL^{\otimes}$ has been shown to be expressively equivalent to the regular product languages and to admit an exponential time decision procedure. In particular, the satisfiability and model checking problems for $DLTL^{\otimes}$ can be solved by product Büchi automata.

The action theory we introduce to model agent communication is an extension of the theory presented in [8], which is based on the logic DLTL, and allows reasoning with incomplete initial states and dealing with postdiction, ramifications as well as with nondeterministic actions. This theory, when extended to the multiagent case [9], allows reasoning about a fixed number of finite state sequential agents, that coordinate their activities by performing their actions together[2].

[2] $DLTL^{\otimes}$ does not allow to describe global properties of a system of agents, since the truth of a formula is evaluated at a state local to an agent and the temporal modalities define causal relationships among local states. The specification of the dynamic of the system is given through the separate specification of the different agents.

In this paper we adapt the theory to the specification and verification of systems of communicating agents, whose behaviour is ruled by an interaction protocol. Following [17, 11] a protocol is defined by describing the effects of communicative actions on the social state, and by specifying the permissions and the commitments that arise as a result of the current conversation state. In our action theory the effects of communicative actions will be modeled by *action laws* which describe the immediate effects produced by an action on the social facts. Permissions, which determine when an action can be taken by an agent, can be modeled by *precondition laws*. Commitment policies, which rule the dynamic of commitments, can be described by *causal laws* which establish the causal dependencies among fluents. The specification of a protocol can be further constrained through the addition of suitable *temporal formulas*, and also the agents' programs can be modeled, by making use of complex actions (regular programs).

The verification that an agent respects commitments and permissions and is compliant with respect to a given protocol, can be formalized as a validity problem in the temporal logic and can be solved by means of temporal model checking. Similarly, the problem of proving protocol properties can be formalized as a validity problem. On the other hand, the problem of determining if an agent is not respecting its social facts at runtime can be modeled as a satisfaction problem (and it can be regarded as an instance of the well known "temporal projection problem").

## 2 The Logic *DLTL* and Its Product Version

In this section first we recall the syntax and semantics of DLTL as introduced in [14], and then recall its product version. $DLTL$ is an extension of LTL in which the next state modality is labelled by actions and the until operator is indexed by programs in Propositional Dynamic Logic (PDL) [12].

Let $\Sigma$ be a finite non-empty alphabet. The members of $\Sigma$ are actions. Let $\Sigma^*$ and $\Sigma^\omega$ be the set of finite and infinite words on $\Sigma$. Let $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. We denote by $\leq$ the usual prefix ordering over $\Sigma^*$ and, for $u \in \Sigma^\infty$, by $prf(u)$ the set of finite prefixes of $u$.

We define the set of programs (regular expressions) $Prg(\Sigma)$ generated by $\Sigma$ as follows: $Prg(\Sigma) ::= a \mid \pi_1 + \pi_2 \mid \pi_1;\pi_2 \mid \pi^*$, where $a \in \Sigma$ and $\pi_1, \pi_2, \pi$ range over $Prg(\Sigma)$. A set of finite words is associated with each program by the mapping $[[]] : Prg(\Sigma) \to 2^{\Sigma^*}$, which is defined in the standard way.

Let $\mathcal{P} = \{p_1, p_2, \ldots\}$ be a countable set of atomic propositions. The set of formulas of DLTL($\Sigma$) is defined as follows:

$$\text{DLTL}(\Sigma) ::= p \mid \neg\alpha \mid \alpha \vee \beta \mid \alpha \mathcal{U}^\pi \beta$$

where $p \in \mathcal{P}$ and $\alpha, \beta$ range over DLTL($\Sigma$) and $\pi$ ranges over $Prg(\Sigma)$.

A model of DLTL($\Sigma$) is a pair $M = (\sigma, V)$ where $\sigma \in \Sigma^\omega$ and $V : prf(\sigma) \to 2^{\mathcal{P}}$ is a valuation function. Given a model $M = (\sigma, V)$, a finite word $\tau \in prf(\sigma)$ and a formula $\alpha$, the satisfiability of a formula $\alpha$ at $\tau$ in $M$, written $M, \tau \models \alpha$, is defined as usual for the classical connectives. Moreover:

- $M, \tau \models p$ iff $p \in V(\tau)$;
- $M, \tau \models \alpha \mathcal{U}^{\pi} \beta$ iff there exists $\tau' \in [[\pi]]$ such that $\tau\tau' \in prf(\sigma)$ and $M, \tau\tau' \models \beta$. Moreover, for every $\tau''$ such that $\varepsilon \leq \tau'' < \tau'$, $M, \tau\tau'' \models \alpha$.

The formula $\alpha \mathcal{U}^{\pi} \beta$ is true at $\tau$ if "$\alpha$ until $\beta$" is true on a finite stretch of behaviour which is in the linear time behaviour of the program $\pi$. A formula $\alpha$ is satisfiable iff there is a model $M = (\sigma, V)$ and a word $\tau \in prf(\sigma)$ such that $M, \tau \models \alpha$.

The derived modalities $\langle \pi \rangle$ and $[\pi]$ can be defined as follows: $\langle \pi \rangle \alpha \equiv \top \mathcal{U}^{\pi} \alpha$ and $[\pi]\alpha \equiv \neg \langle \pi \rangle \neg \alpha$. Furthermore, if we let $\Sigma = \{a_1, \ldots, a_n\}$, then $\bigcirc$ (next), $\Diamond$ and $\Box$ of LTL can be defined as follows: $\bigcirc \alpha \equiv \bigvee_{a \in \Sigma} \langle a \rangle \alpha$, $\Diamond \alpha \equiv \top \mathcal{U}^{\Sigma^*} \alpha$, $\Box \equiv \neg \Diamond \neg \alpha$.

Let us now recall the definition of $DLTL^{\otimes}$ from [13]. Let $Loc = \{1, \ldots, K\}$ be a set of *locations*, the names of the agents synchronizing on common actions. A *distributed alphabet* $\tilde{\Sigma} = \{\Sigma_i\}_{i=1}^{K}$ is a family of (possibly non-disjoint) alphabets, with each $\Sigma_i$ a non-empty, finite set of actions ($\Sigma_i$ is the set of actions which require the participation of agent $i$). Let $\Sigma = \bigcup_{i=1}^{K} \Sigma_i$. For $\sigma \in \Sigma^{\infty}$, we denote by $\sigma \uparrow i$ the projection of $\sigma$ down to $\Sigma_i$.

Atomic propositions are introduced in a local fashion, by introducing a non-empty set of atomic propositions $\mathcal{P}$. For each proposition $p \in \mathcal{P}$ and agent $i \in Loc$, $p_i$ represents the "local" view of the proposition $p$ at $i$, and is evaluated in the local state of agent $i$.

Let us define the set of formulas of $DLTL^{\otimes}(\tilde{\Sigma})$ and their locations (if $\alpha$ is a formula, then $loc(\alpha)$, which is a subset of $Loc$, denotes its location): (a) $\top$ is a formula and $loc(\top)=\emptyset$; (b) if $p \in \mathcal{P}$ and $i \in Loc$, $p_i$ is a formula and $loc(p_i) = \{i\}$; (c) if $\alpha$ and $\beta$ are formulas, then $\neg\alpha$ and $\alpha \vee \beta$ are formulas and $loc(\neg\alpha) = loc(\alpha)$ and $loc(\alpha \vee \beta) = loc(\alpha) \cup loc(\beta)$; (d) if $\alpha$ and $\beta$ are formulas and $loc(\alpha), loc(\beta) \subseteq \{i\}$ and $\pi \in Prg(\Sigma_i)$, then $\alpha \mathcal{U}_i^{\pi} \beta$ is a formula and $loc(\alpha \mathcal{U}_i^{\pi} \beta) = \{i\}$. Notice that no nesting of modalities $\mathcal{U}_i$ and $\mathcal{U}_j$ (for $i \neq j$) is allowed, and the formulas in $DLTL^{\otimes}(\tilde{\Sigma})$ are boolean combinations of formulas from the set $\bigcup_i DLTL_i^{\otimes}(\tilde{\Sigma})$, where

$$DLTL_i^{\otimes}(\tilde{\Sigma}) = \{\alpha \mid \alpha \in DLTL^{\otimes}(\tilde{\Sigma}) \textit{ and } loc(\alpha) \subseteq \{i\}\}.$$

A model of $DLTL^{\otimes}(\tilde{\Sigma})$ is a pair $M = (\sigma, V)$, Where $\sigma \in \Sigma^{\infty}$ and $V = \{V_i\}_{i=1}^{K}$ is a family of functions $V_i$, where $V_i : prf(\sigma \uparrow i) \to 2^{\mathcal{P}}$ is the valuation function for location $i$.

The satisfiability of formulas in a model is defined as above, except that propositions are evaluated locally. In particular, for all $\tau \in prf(\sigma)$:

- $M, \tau \models p_i$ iff $p \in V_i(\tau \uparrow i)$;
- $M, \tau \models \alpha \mathcal{U}_i^{\pi} \beta$ iff there exists a $\tau'$ such that $\tau' \uparrow i \in [[\pi]]$, $\tau\tau' \in prf(\sigma)$ and $M, \tau\tau' \models \beta$. Moreover, for all $\tau'' \in prf(\tau')$, if $\varepsilon \leq \tau'' \uparrow i < \tau' \uparrow i$, then $M, \tau\tau'' \models \alpha$.

Satisfiability in $DLTL^{\otimes}$ is defined as above. Moreover, the derived modalities $\langle\pi\rangle_i$, $[\pi]_i$, $\bigcirc_i$, $\Diamond_i$ and $\Box_i$ are defined as above, but only considering the actions in $\Sigma_i$.

## 3 Action Theories and Systems of Communicating Agents

In this section we introduce the main features of the action theory developed in [8] and (for the multiagent case) in [9]. The behavior of the global system will emerge as the product of the behaviors of the single agents which interact by synchronizing on communicative actions. Each agent participating in the action execution has its own local description of the action determining the action effects on its local state.

The global state of the system can be regarded as a set of local states, one for each agent $i$. The action laws and causal laws of agent $i$ describe how the local state of $i$ changes when an action $a \in \Sigma_i$ is executed. The underlying model of communication is the synchronous one: the communication action $send_{i,j}(m)$ (message $m$ is sent by agent $i$ to agent $j$) is shared by agent $i$ (the sender) and agent $j$ (the receiver) and executed synchronously by them. Their local states are updated separately, according to their action specification. Though, for simplicity, we adopt the synchronous model, an asynchronous model can be easily obtained by explicitly modelling the communication channels among the agents as distinct locations.

The meaning of communicative actions is fixed by the *protocol* which describes the effects of each action on the social state of the system. These effects, including the creation of new commitments, can be expressed by means of *action laws*. Moreover, the protocol establishes a set of preconditions on the executability of actions, which can be expressed by means of *precondition laws*. Each agent has a local view of the social state and the execution of a communicative action can in general affect both the state of the sender and the state of the receiver. In particular, all agents can see the effects on the social state of the actions to which they participate[3]. We assume that the executability of actions is only determined by preconditions on the state of the sender and that a communicative action is always executable for the receiver. Causal and precondition laws are as in [8, 9][4].

As a running example we will use the NetBill protocol, developed for buying and selling goods on the Internet [20].

[3] Observe that, while in the case of a two agents system the history of all communications is known to both agents (as they participate in all communicative actions) and they have the same local view of the social state, this is not true for more than two participants.

[4] Given an action $a$ with precondition $\alpha$, a precondition law has the form $\neg\alpha \rightarrow [a]\bot$, which expresses that the execution of an action $a$ is not possible (i.e. there is no resulting state following the execution of $a$) if $\alpha$ does not hold.

*Example 1.* The protocol begins with a customer requesting a quote for some desired goods, followed by the merchant sending the quote. If the customer accepts the quote, then the merchant delivers the goods and waits for an electronic payment order. After receiving the payment, the merchant forwards the receipt to the customer, who can then successfully decrypt and use the goods. The agents have the freedom to start an exchange at any point in this sequence. For instance, the merchant may begin by sending the quote without waiting for a request; the customer may send an 'accept' message without prior conversation on the price of the goods.

To model this example, we introduce two locations, corresponding to the two agents $mr$, the merchant, and $ct$, the customer, that is $Loc = \{mr, ct\}$. We call *fluents* the atomic propositions in $\mathcal{P}$ indexed by an agent name $i$. Namely, for each $p \in \mathcal{P}$ and $i \in Loc$, $p_i$ is a fluent local to $i$. However, we will omit the index $i$ in $p_i$ when it is clear from the context, for instance, when $p_i$ is in the scope of a modality labeled with $i$ (and we will write $[a]_i p$ for $[a]_i p_i$).

The (domain specific) fluents for this protocol are the following ones: $goods$, $request$, $receipt, paid, accept$. There are also special fluents to represent commitments. They can be base-level commitments, of the form $C(ag_1, ag_2, action)$ (agent $ag_1$ is committed to agent $ag_2$ to execute the $action$), or they can be conditional commitments of the form $CC(ag_1, ag_2, p, action)$ (agent $ag_1$ is committed to agent $ag_2$ to execute $action$, if the condition $p$ is brought out)[5]. In this example, we will use the following fluents to represent commitments: $C(mr, ct, sendGoods)$, $C(mr, ct, sendReceipt)$, $CC(mr, ct, accept, sendGoods)$ and $CC(mr, ct, paid, sendReceipt)$.

In this example, the two agents share exactly the same communicative actions, which are the following ones: $sendQuote$, $sendGoods$, $sendReceipt$ for which the sender is the merchant and the receiver is the customer; $sendRequest$, $sendAccept$ and $sendPayment$, for which the sender is the customer and the receiver is the merchant. These actions are executed synchronously by the sending and receiving agents.

Let us consider the protocol from the point of view of the merchant. The effects of actions on the *merchant's view of the social state* are described by the following action laws:

Actions of the merchant:

(1) $\Box_{mr}([sendQuote]_{mr}(CC(mr, ct, accept, sendGoods) \wedge CC(mr, ct, paid, sendReceipt)))$

(2) $\Box_{mr}(request \rightarrow [sendQuote]_{mr} \neg request)$

(3) $\Box_{mr}([sendGoods]_{mr}(goods \wedge CC(mr, ct, paid, sendReceipt)))$

(4) $\Box_{mr}([sendReceipt]_{mr} receipt)$

[5] The two kinds of base-level and conditional commitments we allow are essentially those introduced in [20]. For simplicity, we do not consider the more general higher level commitments in [17]. Moreover, as in [11] and differently from [20], agents are committed to execute an action rather than to achieve a condition.

Actions of the customer:
(5) $\Box_{mr}([sendRequest]_{mr} request)$
(6) $\Box_{mr}([sendAccept]_{mr} accept)$
(7) $\Box_{mr}([sendPayment]_{mr} paid)$

For instance, the first law says that, when the merchant sends the quote for the good then he commits to send the goods if the customer accepts the request, and he also commits to send the receipt if the customer pays the agreed amount. It is an example of action, whose effect is to create new commitments. The second law says that when the merchant sends the quote for the good when there is a request, then the request is cancelled.

The preconditions on the executability of actions are social facts. We assume they are preconditions on the state of the sender[6], while we assume that it is always possible for an agent to receive a message. The only *precondition law* for the merchant is the following one:

(8) $\Box_{mr}(\neg paid \rightarrow [sendReceipt]_{mr} \bot)$

meaning that when the payment has not been done by the customer, the action *sendReceipt* cannot be executed. All other actions are always executable by the merchant. We will denote by $Perm_i$ the set of precondition laws of agent $i$.

The action laws describing the effects of communicative actions on the state of the customer are exactly the same as for the merchant (as the two agents share all their communicative actions), apart from the fact that all modalities and propositions are indexed with $ct$. We number them (1')–(7'). The only condition on the executability of actions for the customer is the following precondition law:

(9) $\Box_{ct}(\neg goods \rightarrow [sendPayment]_{ct} \bot)$

meaning that the customer may send a payment for the goods only if he has received the goods (all other actions are always executable for the customer).

Let us now define the rules for reasoning about the commitments. In the general case, the actions in the protocol can be regarded as operations on commitments, not only creation operations, but also cancellation, fulfilling and manipulation operations [17]. While it is clear that in this logical formalization it would be simple to define operations to manipulate commitments, in this paper we will assume that all (base-level) commitments of an agent have to be fulfilled for the agent to be compliant with the protocol. Some reasoning rules have to be defined, for cancelling commitments when they have been fulfilled and for dealing with conditional commitments. We introduce the following laws (where $k = i, j$):

(10) $\Box_k([a]_k \neg C(i, j, a))$
(11) $\Box_k([a]_k \neg CC(i, j, p, a))$
(12) $\Box_k((CC(i, j, p, a) \wedge \bigcirc_k p) \rightarrow \bigcirc_k(C(i, j, a) \wedge \neg CC(i, j, p, a)))$

[6] These precondition laws provide a formalization of what are called the PERMIT constraints in [11]. Such permissions do not force the agent to do anything, but determine which actions are executable in a state.

where we assume that the action $a$ is shared by the agents $i$ (debtor) and $j$ (creditor) (if not an explicit *discharge* action should be introduced to make the creditor aware that the commitment has been fulfilled by the debtor). A commitment (or a conditional commitment) to execute an action $a$ is cancelled when the action $a$ has been executed by the debtor (action laws (10) and (11)). A conditional commitment $CC(i, j, p, a)$ becomes a base-level commitment $C(i, j, a)$ when the condition $p$ has been brought out (law (12)). This last law is a *causal law*. Causal laws describe the causal dependencies among fluents and are needed to model the ramification effects of actions [8].

The specification of a system of communicating agents is hence given by defining the protocol for each agent $i$ through a domain description $D_i$. $D_i$ is a tuple $(\Pi_i, \mathcal{C}_i, Perm_i)$ where: $\Pi_i$ is a set of *action laws* and *causal laws*; $\mathcal{C}_i$ is a set of *constraints* and $Perm_i$ is a set of *precondition laws*. The set of constraints $\mathcal{C}_i$ contains all the temporal formulas which might be needed to constrain the behaviour of agent $i$. It includes the *observations* about the value of fluents in different $i$-states, which have the form $[a_1; \ldots; a_j]_i\alpha$, meaning that $\alpha$ holds in the $i$-state obtained after executing the action sequence $a_1, \ldots, a_j$[7]. Though also the precondition laws in $Perm_i$ are constraints, we have kept them separate from $\mathcal{C}_i$, as they have a special role in the verification of the agent system. A *(distributed) domain description* $D$ is a family of domain descriptions $D_i$, one for each agent $i$.

Observe that action laws and causal laws only describe the changes to the state. All other fluents which are not changed by the actions are assumed to persist unaltered to the next state. They are called *frame* fluents. To cope with the frame problem, the laws in $\Pi_i$, describing the (immediate and ramification) effects of actions, have to be distinguished from the constraints in $\mathcal{C}_i$ and given a special treatment. In [8], to deal with the frame problem, a completion construction is defined which, given a domain description, introduces frame axioms for all the frame fluents in the style of the successor state axioms introduced by Reiter [16] in the context of the situation calculus. The completion construction is applied only to the action laws and causal laws in $\Pi_i$ and not to the constraints. In the following we call $Comp(\Pi_i)$ the completion of a set of laws $\Pi_i$ (the action laws and causal rules of each agent $i$ have to be completed separately) and we refer to [8] for the details on the completion construction.

Given a distributed domain description $D$, let the completed domain description $Comp(D)$ be the set of formulas $\bigwedge_j (Comp(\Pi_j) \wedge \mathcal{C}_j \wedge Perm_j)$. The runs of the system according the protocol are the linear models of $Comp(D)$: in a model $(\sigma, V)$, the behaviour of each agent $i$ is a subsequence $\sigma_i$ of the system behaviour $\sigma$. Observe that these protocol runs may still contain pending commitments, that is commitments which have not been fulfilled.

In the next section we will address, among others, the problem of verifying the compliance of an agent with respect to a protocol. In our formalism the program

[7] For instance, in the example above we can assume that all fluents are false in the initial state, i.e. $\mathcal{C}_i = \{\neg f_k, \text{ for all fluents } f_k\}$

of an agent can be specified by making use of complex actions. Consider the following program $\pi_{mr}$ for the merchant:

$$[\neg done?; ((sendRequest; sendQuote) + (sendAccept; sendGoods) + (sendPayment; sendReceipt; exit))]^*; done?$$

The program above describes the behaviour of a reactive agent which sends the quote when receiving a request, sends the goods when receiving the "accept", sends a receipt after receiving the payment and terminates after sending the receipt. We assume that the fluent *done* is initially false and that the action *exit* has the only effect of making it true (i.e. $\Box_{mr}[exit]_{mr}done$). In general, the *program of an agent i* is specified by a domain description $Prog_i = (\Pi_i^p, \mathcal{C}_i^p)$. In the example above, we have: $\Pi_{mr}^p = \{\Box_{mr}[exit]_{mr}done\}$ and $\mathcal{C}_{mr}^p = \{\langle \pi_{mr} \rangle_i \top, \neg done\}$. Observe that $\Pi_{mr}^p$ contains the specification of those actions which are local to the agent *mr* (in this case only the action *exit*). The constraints in $\mathcal{C}_{mr}^p$ say that in the initial state the program $\pi_{mr}$ for the merchant is executable and the fluent *done* is is false.

The action *done*? is a test action, in the style of dynamic logic. Test actions allow the choice among different behaviours to be controlled. As $DLTL^{\otimes}$ does not include test actions, we introduce them in the language as atomic actions in the same way as done in [8]. To allow agent $i$ to test the value of a proposition $\phi$ in its local state, we introduce the modality $[\phi?]_i$ (regarded as an atomic action in $\Sigma_i$). We assume that, for all test actions occurring in a domain description, the corresponding action laws are implicitly added.

## 4 Verification of Communicating Agents

In this section we consider some different types of verification problems that can be solved in this framework. As in [11] we consider the following four types of verification for an open system:

1. Verify that an agent will always satisfy its social facts;
2. Verify the outcome of a system, assuming unknown agents are compliant;
3. Prove a property of a protocol;
4. Determine if an agent is not respecting the social facts at runtime.

While the fourth property is a run-time property, the first three can be verified at design time.

For the first verification problem, we assume that we have access to agent $i$ program code, $Prog_i$. Then at design time we can verify that this agent respects its permissions and its commitments. We say that an agent $i$ *satisfies its permissions* if, in all runs of the system, when a communicative action $a$ is executed, then its execution is possible according to the precondition laws in $Perm_i$, that is, all precondition laws $\Box(\alpha \rightarrow [a]\bot) \in Perm_i$ are satisfied. An agent $i$ *satisfies its commitments* when, in all the runs of the system, for all the possible commitments $C(i, j, a)$ in which agent $i$ is the debtor, the formula

$$\Box_i(C(i, j, a) \rightarrow \Diamond_i \langle a \rangle_i \top)$$

holds. Such a formula says that, when an agent is committed to execute action $a$, then it must eventually execute $a$. We denote with $Com_i$ the set of all the formulas, as the one above, describing the satisfaction of the commitments of agent $i$.

To formalize the first verification problem assume that the specification of the program of agent $i$ is given by the domain description $Prog_i = (\Pi_i^p, \mathcal{C}_i^p)$, and the protocol is given by the distributed domain description $D$, where the domain description for each agent $j$ in $Loc$ is $D_j = (\Pi_j, \mathcal{C}_j, Perm_j)$. The verification problem 1 can be formalized as a validity check. The formula $(\bigwedge_j(Comp(\Pi_j) \wedge \mathcal{C}_j) \wedge \bigwedge_{j \neq i}(Perm_j \wedge Com_j) \wedge Comp(\Pi_i^p) \wedge \mathcal{C}_i^p) \rightarrow (Perm_i \wedge Com_i)$ (where $j$ ranges on all agents) is valid if in all the behaviours of the system, in which agent $i$ respects its specification $Prog_i$ and all other agents $j$ (whose internal program is unknown) respect the protocol specification (including their permissions and commitments), the permissions and commitment of agent $i$ are also satisfied. This means that agent $i$ is compliant (respects its "social facts") under the assumption that all other agents in the protocol are compliant[8].

The verification problem 2 can be formalized similarly. Again we only have information about the behaviour of agent $i$ and we want to show that the outcome $\alpha$ of the system will eventually be achieved. The kind of properties that can be captured in this framework have to refer to the local states of the different agents separately (there is no global state). For instance, we can require that agents $h$ and $i$ will eventually reach the states $\alpha_h$ and $\alpha_i$. To check this, we can check the validity of the formula: $(\bigwedge_j(Comp(\Pi_j) \wedge \mathcal{C}_j) \wedge \bigwedge_{j \neq i}(Perm_j \wedge Com_j) \wedge Comp(\Pi_i^p) \wedge \mathcal{C}_i^p) \rightarrow (\Diamond_h \alpha_h \wedge \Diamond_i \alpha_i)$. Observe that here we have not made any assumption that agent $i$ is compliant, we just assume that it behaves according to its program. The other agents, instead, are assumed to be compliant.

In the verification problem 3, the internals of all agents are not accessible. We only know that the agents are compliant with the protocol and we want to prove some property $p$ of the protocol. Again we can formalize it as a validity check of the formula: $(\bigwedge_j(Comp(\Pi_j) \wedge \mathcal{C}_j) \wedge \bigwedge_j(Perm_j \wedge Com_j)) \rightarrow p$.

The verification problem 4 is performed at run-time. Nothing is known about the internals of the agents. An external observer observes the history of the communications among the agents up to one state (that is the sequence of communicative actions that have been executed). Given such a history we want to check if there is evidence that some agent is not compliant. In essence, we want to verify if the history is a possible (prefix of) an allowed behaviour of the system, that is a behaviour in which all agents are compliant with the protocol (if not, the history gives evidence that some agent violates the protocol). This can be formalized as a satisfiability problem. Let $\tau = a_1, \ldots, a_n$ be the observed actions sequence. Let $\tau \uparrow 1, \ldots, \tau \uparrow K$ be the projections of the action sequence $\tau$ on the actions of the agents $\{1, \ldots, K\}$, and let $\tau \uparrow i = a_1^i, \ldots, a_{n_i}^i$. The formula:

[8] Observe that while the behaviours of agent $i$ are constrained both by the protocol and by its own specification, other agents may perform any action sequence which is compatible with the constraints posed by the protocol on their behaviour and the fact that they have to fulfill their commitments.

$\bigwedge_j(Comp(\Pi_j)\wedge\mathcal{C}_j)\wedge\bigwedge_j(Perm_j\wedge Com_j)\wedge(\langle a_1^1;\ldots;a_{n_1}^1\rangle\top\wedge\ldots\wedge\langle a_1^K;\ldots;a_{n_K}^K\rangle\top)$ will be satisfied if there is a behaviour $\sigma$ of the system (a model $(\sigma, V)$) in which, for each $i = 1\ldots N$, the behaviour of agent $i$ starts with the action sequence $\tau\uparrow i = a_1^i,\ldots,a_{n_i}^i$.

*Example 2.* We provide an example of the verification problem 1 to verify that the program of the merchant, specified by the domain description $Prog_{mr} = (\Pi_{mr}^p, \mathcal{C}_{mr}^p)$ above, satisfies the social facts. Let

$\Pi_{mr} = \{(1),\ldots,(7),(10),(11),(12)\}$, $\mathcal{C}_{mr} = \{\neg f_{mr} : \forall\text{fluents } f_{mr}\}$,
$Perm_{mr} = \{(8)\}$
$\Pi_{ct} = \{(1'),\ldots,(7'),(10),(11),(12)\}$, $\mathcal{C}_{ct} = \{\neg f_{ct} : \forall\text{fluents } f_{ct}\}$,
$Perm_{ct} = \{(9)\}$

$Com_{mr}$ contains the formulas

$\Box_{mr}(C(mr, ct, sendGoods) \rightarrow \Diamond_{mr}\langle sendGoods\rangle_{mr}\top)$
$\Box_{mr}(C(mr, ct, sendReceipt) \rightarrow \Diamond_{mr}\langle sendReceipt\rangle_{mr}\top)$

and $Com_{ct} = \emptyset$. The formula:

$$Comp(\Pi_{mr}) \wedge \mathcal{C}_{mr} \wedge Comp(\Pi_{ct}) \wedge \mathcal{C}_{ct} \wedge Perm_{ct} \wedge Com_{ct}\wedge \\ \wedge Comp(\Pi_{mr}^p) \wedge \mathcal{C}_{mr}^p \rightarrow (Perm_{mr} \wedge Com_{mr})$$

is valid if in all the behaviours of the system in which the merchant respects its specification $Prog_{mr}$ and the customer (whose internal program is unknown) respects its permissions and commitments, the permissions and commitments of the merchant are also satisfied.

The above verification and satisfiability problems can be solved by extending the standard approach for verification and model-checking of Linear Time Temporal Logic, based on the use of Büchi automata. As described in [14], the satisfiability problem for DLTL can be solved in deterministic exponential time, as for LTL, by constructing for each formula $\alpha \in DLTL(\Sigma)$ a Büchi automaton $\mathcal{B}_\alpha$ such that the language of $\omega$-words accepted by $\mathcal{B}_\alpha$ is non-empty if and only if $\alpha$ is satisfiable. This result has been extended in [13] to $DLTL^\otimes$. The verification of a formula $\alpha \rightarrow \beta$ can be carried out by first constructing the two Büchi automata for $\alpha$ and the *negation* of $\beta$. If the two automata have a common execution sequence, this sequence provides a counterexample for $\alpha \rightarrow \beta$. Thus $\alpha \rightarrow \beta$ is valid if the language accepted by the product of the two automata is empty.

The similarities between the verification approach for LTL and that for $DLTL^\otimes$ suggest the possibility of using techniques and tools which have been developed for LTL. For instance, it is possible to extend to $DLTL^\otimes$ the efficient tableau-based algorithm of [6] for constructing the automaton on the fly.

## 5 Conclusions

We have shown that $DLTL^{\otimes}$ is a suitable formalism for specifying and verifying a system of communicating agents. The issue of developing semantics for agent communication languages has been examined in [18], by considering in particular the problem of giving a *verifiable* semantics, i.e. a semantics *grounded* on the computational models.

Guerin's thesis [11] defines an agent communication framework which gives agent communication a grounded declarative semantics. In particular it has different languages for agent programming, for specifying agent communication and social facts, and for expressing temporal properties. Our approach provides a unified framework for describing different aspects of multi-agent systems using $DLTL^{\otimes}$. Programs are expressed as regular expressions, (communicative) actions can be specified by means of action and precondition laws, properties of social facts can be specified by means of causal laws and constraints, and temporal properties can be expressed by means of the *until* operator.

While in this paper we follow a social approach to the specification and verification systems of communicating agents, [2] and [19] have adopted a mentalistic approach. The goal of [2] is to extend model checking to make it applicable to multi-agent systems, where agents have BDI attitudes. This is achieved by using a new logic which is the composition of two logics, one formalizing temporal evolution and the other formalizing BDI attitudes. In [19] agents are written in MABLE, an imperative programming language, and have a mental state. MABLE systems may be augmented by the addition of formal claims about the system, expressed using a quantified, linear time temporal BDI logic. MABLE makes use of the SPIN model checker to automatically verify the claims. The paper does not deal with the problem of proving properties of protocols. The same can be said for the paper by Yolum and Singh [20], which presents a social approach based on event calculus to protocol specification and execution.

A related approach is that of ConGolog [5], an extended version of the language Golog, that incorporates a rich account of concurrency, in which complex actions (plans) can be formalized as Algol-like programs in the situation calculus. A substantial difference with ConGolog, apart from the different logical foundation, is that here we model agents with their own local states, while in Congolog the agents share a common global environment and all the properties are referred to a global state.

## References

[1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. in *Annals of Mathematics and AI*, 22:5–27, 1998. 263

[2] M. Benerecetti, F. Giunchiglia and L. Serafini. Model Checking Multiagent Systems. *Journal of Logic and Computation*. Special Issue on Computational Aspects of Multi-Agent Systems, 8(3):401-423. 1998. 273

[3] D. Calvanese, G. De Giacomo and M. Y.Vardi. Reasoning about Actions and Planning in LTL Action Theories. In Proc. *KR'02*, 2002. 263

[4] P. Cohen and H. Levesque, Communicative actions for artificial agents. *Int. Conf. on Multi Agent Systems*, pp.65-72, MIT Press,1995. 262

[5] G. De Giacomo, Y. Lespérance, H. J. Levesque. ConGolog, a concurrent programing language based on the situation calculus. *Artificial Intelligence* 121(2000), pp.109-169. 273

[6] R. Gerth and D. Peled and M. Y.Vardi and P. Wolper. Simple On-the-fly Automatic verification of Linear Temporal Logic. In *Proc. 15th Work. Protocol Specification, Testing and Verification*, Warsaw, June 1995, North Holland. 272

[7] L. Giordano, A. Martelli, and C. Schwind. Ramification and causality in a modal action logic. In *Journal of Logic and Computation*, 10(5):625-662, 2000.

[8] L. Giordano, A. Martelli, and C. Schwind. Reasoning About Actions in Dynamic Linear Time Temporal Logic. In FAPR'00 - Int. Conf. on Pure and Applied Practical Reasoning, London, September 2000. Also in *The Logic Journal of the IGPL*, Vol. 9, No. 2, pp. 289-303, March 2001. 262, 263, 266, 269, 270

[9] L. Giordano, A. Martelli, and C. Schwind. Reasoning about Actions in a Multiagent Domain. In *Proc. AI*IA'01*, Bari, pp. 237–248, September 2001. 263, 266

[10] F. Giunchiglia and P. Traverso. Planning as Model Checking. In *Proc. The 5th European Conf. on Planning (ECP'99)*, pp.1–20, Durham (UK), 1999. 263

[11] F. Guerin. Specifying Agent Communication Languages. PhD Thesis, Imperial College, London, April 2002. 262, 264, 267, 268, 270, 273

[12] D. Harel. First order dynamic logic in *Extensions of Classical Logic, Handbook of Philosophical Logic II*, pp. 497–604, 1984. 264

[13] J. G. Henriksen and P. S. Thiagarajan A product Version of Dynamic Linear Time Temporal Logic. in *CONCUR'97*, 1997. 263, 265, 272

[14] J. G. Henriksen and P. S. Thiagarajan Dynamic Linear Time Temporal Logic. in *Annals of Pure and Applied logic*, vol.96, n.1-3, pp.187–207, 1999 264, 272

[15] Y. Labrou and T. Finin, A semantic approach for KQML - a general purpose communication language for software agents. In *3rd Int Conf. on Information and Knowledge Management, CIKM'94*, pp.447-455, 1994. 262

[16] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, V. Lifschitz, ed.,pages 359–380, Academic Press, 1991. 269

[17] M. P. Singh. A social semantics for Agent Communication Languages. In *IJCAI-98 Workshop on Agent Communication Languages*, Springer, Berlin, 2000. 262, 264, 267, 268

[18] M. Wooldridge. Semantic Issues in the Verification of Agent Communication Languages. *Autonomous Agents and Multi-Agent Systems*, vol. 3, pp. 9-31, 2000. 273

[19] M. Wooldridge, M. Fisher, M. P. Huget and S. Parsons. Model Checking Multi-Agent Systems with MABLE. In *AAMAS'02*, pp. 952–959, Bologna, Italy, 2002. 273

[20] P. Yolum and M. P. Singh. Flexible Protocol Specification and Execution: Applying Event Calculus Planning using Commitments. In *AAMAS'02*, pp. 527–534, Bologna, Italy, 2002. 266, 267, 273

# Combining Intelligent Agents and Animation

Adam Szarowicz and Peter Forte

School of Computing and Information Systems, Kingston University
Penrhyn Road, Kingston-upon-Thames, KT1 2EE, UK
tel. +44 20 8547 2000 ext. 62142
{a.szarowicz,pforte}@kingston.ac.uk

**Abstract.** This paper reviews FreeWill and other AI architectures that have contributed to the ideas, which underlie it. "FreeWill" proposes and implements a cognitive architecture designed for easy creation of animated scenes with many autonomous agents interacting in various ways. The generation of animated human figures especially in crowd scenes has many applications in such domains as the special effects industry, computer games or for the simulation of the evacuation from crowded areas. Currently such scenes have to be created by human animators using dedicated software packages. This is both expensive and time-consuming. FreeWill combines and extends two existing architectures - Funge's cognitive architecture and the recent SAC concept. Additionally it puts emphasis on strong design and provides easy co-operation of different software packages.

**Keywords:** Intelligent agents, AI architectures, lifelike characters

## 1 Introduction

The creation of animated scenes involving interacting characters is a problem in such applications as film post-production and special effects, computer games or event simulation in crowded areas. Ideally, animated characters should have some autonomy and awareness of their environment, which is a feature lacking in cur-rent animation packages. We address this problem by proposing an agent-based cognitive architecture FreeWill, which can be applied to extend modern animation packages. The ultimate goal of our research is to propose an animation-dedicated framework, which would increase a character's autonomy and relieve the animator from the most tedious part of the process.

Despite the benefits to be gained in computer generated animation from such autonomous agents, their use is still at a restricted and rudimentary stage. There is a technical reason for this: the AI architectures needed to underpin their performance do not yet exhibit the right balance of features regarding functionality, efficiency and ease of software engineering. In this paper we review a number of existing AI architectures and assess their suitability for the task. This is followed by our own proposals based on ideas from Funge (Funge, 1999) and Winikoff (Winikoff *et al*, 2001). The AI architecture we propose is also coupled

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 275–286, 2003.
© Springer-Verlag Berlin Heidelberg 2003

to a corresponding and clear software engineering architecture expressed in the Unified Modeling Language (UML). We argue that this represents the correct direction for future AI architectures aimed at supporting autonomous intelligent agents in computer assisted animation. Finally we describe a prototype we have implemented to illustrate the practicality of the proposal.

## 2 Existing Architectures

In recent years we have been observing a substantial increase in graphics processing power especially with the appearance of cheap hardware graphic accelerators. This has improved the quality of computer graphics available in home and office applications such as CAD programs or computer games. It has also boosted the level of special effects in film post-production. However a noticeable gap exists between the possible quality of character animation and the level of intelligence in which the characters are equipped by game developers or even professional animation packages. Despite several attempts to make the animated characters more intelligent (Funge, 1999, Yoon *et al* 2000, Isla *et al* 2001, Flake *et al*, 2001, Blumberg *et al* 2002) there is still a broad scope for research into the problems involved in automatic creation of computer characters. The growing popularity of agent-based architectures and methodologies (Rao and Georgeff, 1995, Wooldridge and Jennings, 2000, Wood and DeLoach, 2001, Mylopoulos *et al*, 2001, Winikoff *et al*, 2001) gives a new opportunity to apply recent advances in AI to this problem. We begin by reviewing a number of candidate AI architec-tures, which could be applied to the task.

For example Soar (Laird *et al*, 1987) is "an architecture for general intelligence" (Newell, 1990). Over the years it has become a very popular and widely used tool allowing many researchers to investigate the working of human cognition, learn more about man-machine interaction and explore other AI-related fields. However due to the uniformity of Soar's design all external knowledge must be first well understood and encoded into Soar production rules (cf. Wray 2002), which can be a tedious and demanding process. For example in the Soar-based real-time system directing computer simulated aircraft pilots (TicAir-Soar) created at the University of Michigan (Laird and Jones, 1998) there were approximately 5200 rules (and over 450 operators), many of them acquired after extensive consultations with experts in the field (military pilots). For a system directing interacting animated characters the developers would have to create hundreds of symbolic rules showing the Soar engine how to handle many 'obvious' things such as biomechanics, inverse kinematics, or world geometry. Additionally Soar does not offer any way of handling multi-agent interactions (or interactions between different instances of the Soar engine) other than through the environment. While this is sufficient to build a successful system the developer would again have to resort to creation of numerous detailed production rules. Therefore it appears that Soar may not be the best tool to create the proposed multi-agent system.

Another alternative we looked at is C4. This is an architecture recently proposed by a research group working at the MIT Media Lab (Isla *et al*, 2001) de-signed to allow creation of autonomous and semi-autonomous creatures. The work is informed by the behaviour of living animals (in particular dogs and wolves). The concept is based on a 'brain' divided into numerous systems communicating through an internal blackboard. The modules making up C4 are: sensory and perception system, action system, navigation and motor system (which in turn is also composed of different layers), short-term memory and the mental blackboard. Thus the whole system is highly modular which is its main architectural principle. The authors of C4 put a strong emphasis on learning especially in the form of training a home pet (a dog) which makes the system biased towards solutions based on this type of interaction. It is not obvious how this design would fit into simulation of interacting characters and whether the learning system is flexible enough to allow unsupervised learning of more advanced activities. Additionally the intercharacter interaction in C4 is limited to flocking behaviour and issue of commands. Therefore some serious changes to the system would probably be necessary in order to allow it to generate many intelligent human-like characters.

A third possibility is the cognitive architecture for animated characters proposed by J. Funge. Funge defines an autonomous character as "a character that, during the course of a computer game or animation, can decide how to behave on its own." (Funge, 1999). He proposes that virtual characters should maintain a cognitive model that is an internal model of their world and additionally an explicit representation of some other knowledge about the character's world. Therefore autonomous cognitive characters have some domain knowledge - for example knowledge about the world's dynamics. At the same time he claims that generally the character should not be omniscient, which happens when the characters gain access to the 'true' world model. The ideas are formalised using the situation calculus.

Funge emphasises that the cognitive layer is only one among many models underlying an animated character (Fig. 1). He recognises that a developer trying to build a virtual scene will be faced with many problems apart from those of artificial intelligence. What he proposes in not only a cognitive architecture but also a whole process of building a virtual character.

Unfortunately the character interaction component seems to be omitted in the model, which concentrates on building a single character without addressing the problem of how to exchange information or co-ordinate behaviour with other creatures. Also the process of building the whole system, including how to merge all the different aspects of the cognitive engine, is not addressed in detail. However it is true to say that a number of extensions would make this architecture very suitable for design and implementation of intelligent characters inhabiting virtual worlds.

Another strong candidate is found in recent proposals by Winikoff (Winikoff *et al*, 2001, Harland and Winikoff, 2001) which extend the BDI architecture (Rao and Georgeff, 1991, Rao and Georgeff, 1995) to make it more accessible as

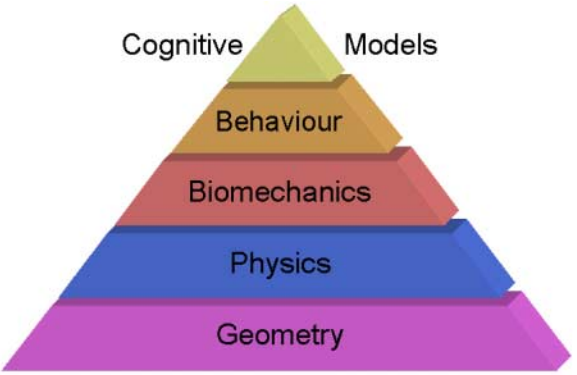



**Fig. 1.** Pyramid depicting different models used to build a virtual character (after Funge, 1999)

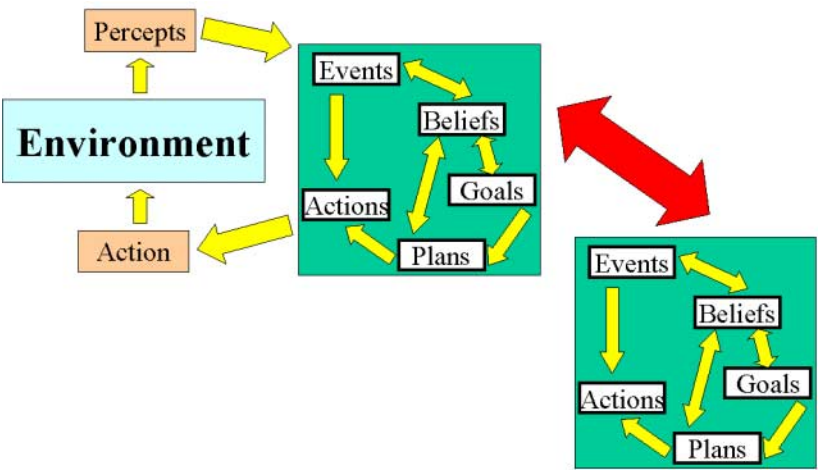



**Fig. 2.** Components of the SAC architecture (after Winikoff, 2001)

a software engineering development framework (see Fig. 2). Winikoff's objective is to propose "a simplified model which retains the power (and efficiency) of the BDI model but allows more people to develop intelligent agent systems" - hence the name SAC (Simplified Agent Concepts). Winikoff's concepts are much more 'implementable' than the original BDI architecture and he puts more emphasis on goals, events and construction of plans.

SAC agent's execution model can be described in 7 steps:

- percepts get interpreted and give rise to events
- agent's beliefs are in turn updated
- events can create reflexive actions and possibly new goals
- goals get updated

– plan is chosen (if necessary)
– chosen plan is expanded
– an action is chosen, scheduled and performed

The developer is responsible for the identification of all the agent's goals - "it is important for the developer to identify the top level goals of the agent as well as subsidiary goals which are used in achieving main goals." (Winikoff et al, 2001) Additionally the agent may be equipped with a plan library from which it can se-lect appropriate plans when necessary.

Other systems we have looked at include Gaia (Wooldridge and Jennings, 2000), Tropos (Mylopoulos *et al*, 2001) MaSE (Wood and DeLoach, 2001), Prometheus (Padgham and Winikoff, 2002) and various notations proposed to design agent-based systems, such as AUML (Bauer *et al*, 2001). However most of these fall into the category of methodologies rather than architectures.

Our conclusion is that of the existing architectures we have considered, two are most suitable for consideration. These are Funge's cognitive architecture (FCA) which addresses many problems closely related to creation of intelligent animated creatures and the SAC architecture proposed to make agent concepts more ap-proachable to software developers. We believe that by merging and extending the ideas which underlie these approaches, more robust and flexible architecture can be created.

## 3 Comparison of the FCA and SAC Architectures

In this section we compare specific features of the FCA architecture proposed by Funge and the agent-based SAC concept before going on to discuss how a synthesis and extension of the two might be achieved.

Both architectures stress the importance of goals, reactive and proactive behaviour, and the need for some representation of the character's knowledge (be-liefs). They are also built based on the sense-think-act cycle - a standard approach to building systems which have to communicate with the environment. But whereas in FCA the sensing process always updates the character's world model in SAC it first gives rise to internal events, which may in turn update the agent's beliefs.

FCA is a dedicated animation architecture, SAC is a general-purpose model for designing and implementing situated agents. What follows from this is that only FCA includes animation concepts such as geometry or inverse kinematics. FCA also addresses the problem of learning in the form that it may be necessary for some characters to be able to learn new behaviour.

Following the BDI's statement that agents should in general maintain a list of predefined plans SAC includes the concept of plan libraries. On the other hand FCA allows to pre-define behaviour (in the form of reactive behaviour rules or state machines) but for the goal-directed behaviour recommends doing searches in (perhaps pruned) situation trees. Similarly the interaction aspect of SAC remains in strong contrast with FCA which does not provide any explicit mechanism to allow interaction between the animated characters (a possibility

**Table 1.** Comparison of the architectures

| **Architecture name** | **FCA** | **SAC** |
|---|---|---|
| Architecture orientation | Character oriented | Agent oriented |
| Recognition for the animation aspect | Yes | No |
| Representation of the character's knowledge | Internal world model + domain knowledge | Beliefs (a knowledge base) |
| Autonomy | Yes | Yes |
| Learning | Recognition for learning | No explicit emphasis on learning |
| Reactive and pro-active behaviour | Yes | Yes |
| Interaction with other characters | Implicit | Explicit |
| Main implementation cycle | Sense-think-act | Sense-think-act |
| Sensing/perceiving | Updates the world model | Gives rise to (internal) events |
| Planning | Search on atomic actions | Plan libraries |
| Ease of implementation | Not obvious | Easy to follow development model |
| Goals | Yes | Yes |
| Formalism | Situation calculus | Under development |

of using a communication protocol is only mentioned). The table below provides a summary of these and other important features of both systems.

## 4 The FreeWill Architecture

In this section we propose our own architecture (FreeWill) building on ideas from the above.

FreeWill's main aim is to propose an architecture suitable to create intelligent and realistic animation in the form of crowd scenes. Therefore it contains elements found in both animation-driven systems and distributed (multiagent) solutions. Each avatar participating in the animation consists of an intelligent agent imple-mented as a modified SAC agent together with a body layer, which is responsible for handling the visible part of the agent (see Fig. 3.).

The geometry, physics (kinematics) and behavioural layers are incorporated from the FCA architecture. Each avatar is ultimately modelled as a set of simple geometrical shapes controlled by the inversed kinematics (supplied by an existing animation package and only visible after rendering). All the characters follow some simple patterns of behaviour, which allow them to be identified as human-like creatures. Additionally we agree with Funge's view that substantial part of the character's knowledge is the internal world model, which is updated by

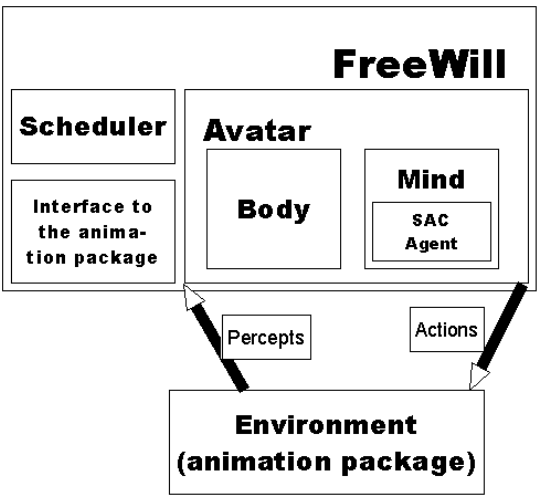



**Fig. 3.** The FreeWill avatar

sensing. On the other hand agent concepts allow us to build the 'mind' - the agents store plan libraries instead of creating plans from scratch every time a new plan is needed, new plans are constructed only if there is no existing template, and of course the characters have to interact with each other. The interaction is fully modelled by reaction to events and as yet doesn't take place through any communication language. Indeed the only use of such a language for this system would be in actions involving direct collaborative interaction between avatars (e.g. a common attempt to lift an object). In other situations the characters should not (because of distance) or need not (handshake will look most natural if synchronised only by means of visual sensors) communicate in a high level language. Such communication might be attempted when the system is sufficiently robust to allow for collaborative tasks between the agents.

The agent components are presented below (Fig. 4).

The main difference between the SAC model and FreeWill is that events to which the agent responds are not created inside the agent but are external - come from the scheduler. They are however submitted there by the agent when it wants to update its knowledge base or when an action must be performed. External events allow for a greater uniformity of the architecture - both acting and sensing events can be modelled in this way and similarly scene objects can be handled following this scheme. It is also in line with the principle of easy implementation. Event-based systems with external events are popular and easy to implement.

The scheduler serves as an event dispatcher and sequencer. All events are man-aged globally, scheduled and then distributed to appropriate avatars. This allows for greater flexibility and also provides a uniform way of both maintaining

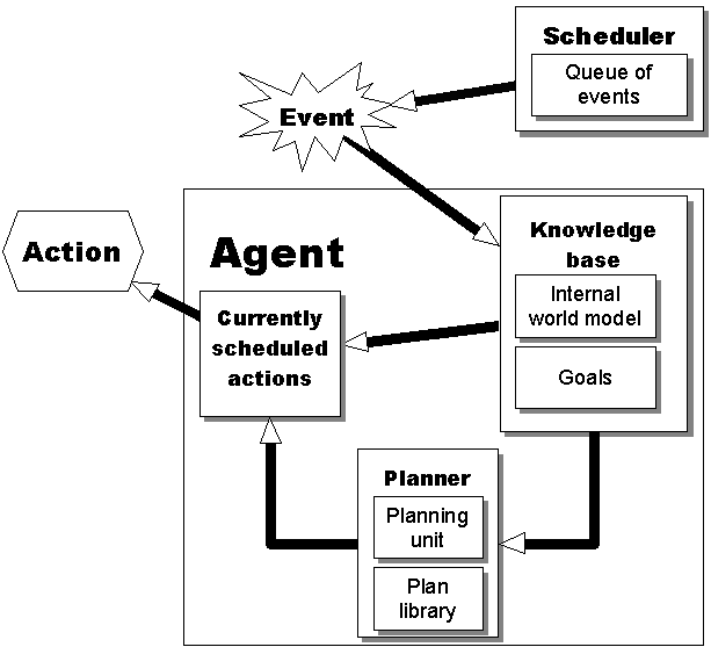



**Fig. 4.** The agent component of the avatar

the avatar's awareness and communicating with other avatars and possibly world ob-jects too. FreeWill execution cycle proceeds as follows:

- a sensing action is executed by the avatar
- agent's beliefs get updated
- current plan is evaluated, if there is a need to perform a reflexive action the current action queue is cancelled and a new action gets submitted
- goals get updated if necessary
- plan is updated if necessary
- last action is chosen and submitted to the scheduler, when the avatar is asked to execute the action it also submits a new sensing action to be put in the event queue (Fig. 4)

Main features of the proposed architecture are gathered in Table 2:

## 5 Implementation and Results

The architecture described above has been modelled using the Unified Modelling Language (for more details see Forte and Szarowicz, 2002, Szarowicz *et al*, 2001) and implemented as an off-line prototype extension to a modern animation package 3D Studio Max. Generation of the solution to a file takes only a few min-utes, the most computationally expensive part is rendering of the animation

**Table 2.** Attributes of the FreeWill architecture

| **Architecture name** | **FreeWill** |
|---|---|
| Architecture orientation | Character oriented |
| Recognition for the animation aspect | Yes |
| Representation of the character's knowledge | Internal world model + knowledge base |
| Autonomy | Yes |
| Learning | Recognition for learning (under implementation) |
| Reactive and pro-active behaviour | Yes |
| Interaction with other characters | Implicit |
| Main implementation cycle | Sense-think-act with external events |
| Sensing/perceiving | Updates the world model |
| Planning | Plan libraries + construction of plans |
| Ease of implementation | Easy to follow (UML documentation exists) |
| Goals | Yes |
| Formalism | None |

(takes hours to days). An example sequence from the animation is given below (Fig. 5). The animation depicts two avatars that know each other ('friends'). The avatars are given primary goals and get placed by the user in desired starting positions. Next the simulation is started and follows the cycle described in the previous section. The avatars find themselves at a distance from each other (so they cannot see the other one) and their primary goal is 'get to the position (X,Y)' (shots 1-2). The avatars start walking and as a result the distance between them decreases (shots 3-4), eventually they recognise each other as a friend and decide to replace the main goal with an intermediate one ('shake hands with a friend'). Their plans get updated and they start the synchronisation sequence preparing for the handshake (shot 5). Next if there is no unexpected problems the handshake is rolled out (shots 6-8) and the avatars return to their original goal - they try to continue their journey to the desired location. However at this point they detect an avatar colliding with their chosen path so a reflexive action is submitted to the action queue ('avoid'), meaning all other actions are cancelled (shots 9-11). Eventually the avatars re-plan a new path to the goal and decide to pursue it (shot 12).

The implemented system was able to successfully generate a scene with 20 avatars walking on a sidewalk, the avatars performed both reactive (e.g. collision avoidance) and proactive (the only implemented one was handshake) actions. Several problems have been identified during the simulation, such as:

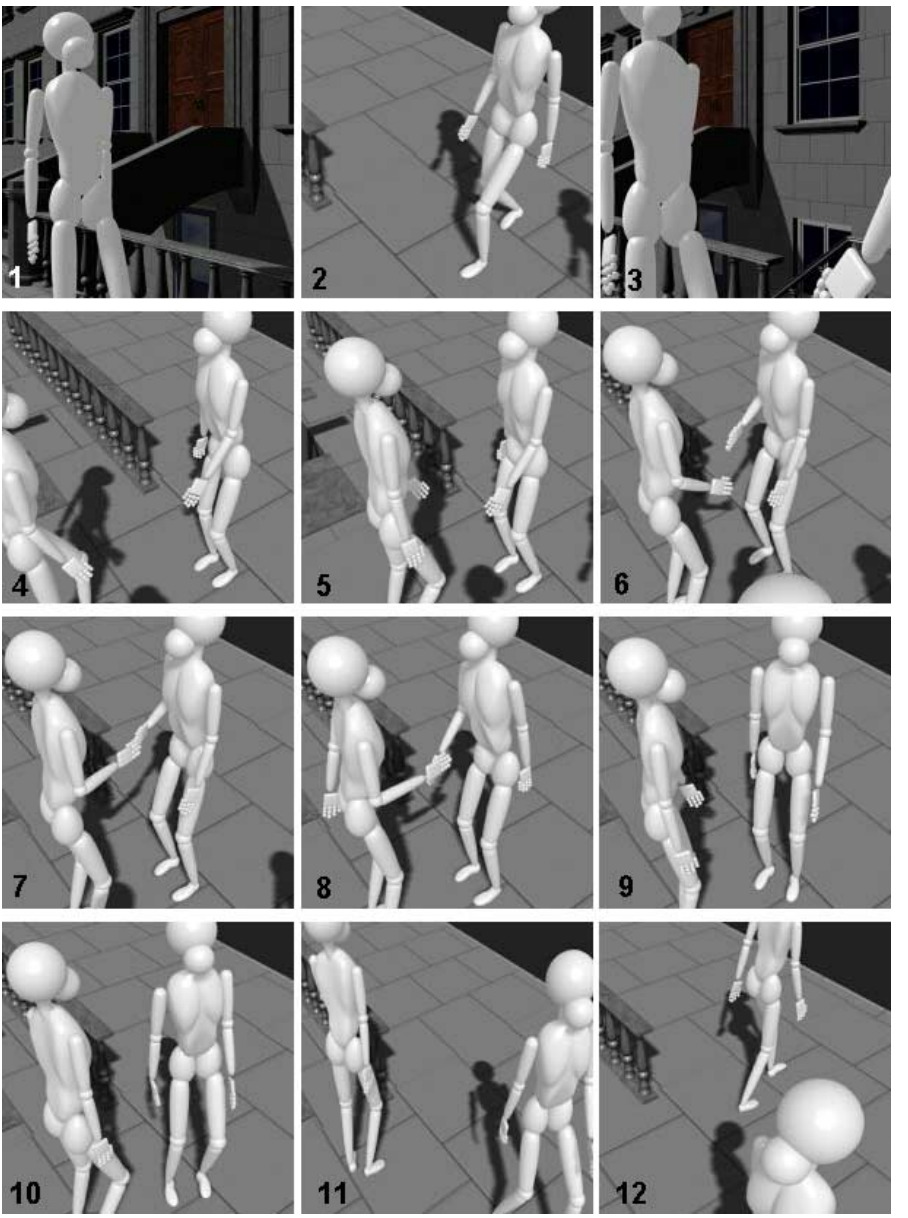



**Fig. 5.** Avatars shaking hands as a result of changing their goal to 'shake hands with a friend'

- The chosen mode of communication with the animation package did not allow for fine control. As a result some of the actions were not performed with a sufficient accuracy. This has now been changed and the system al-lows for a much higher level of precision.
- In order to create more varied and interesting animations it is necessary to create many more proactive high-level actions (such as a handshake), which could then be stored in the plan library. This can be done both manually (hand-made animations - handshake animation is one example) or automatically. A module incorporating machine learning is now being developed to fulfil the latter.

## 6 Future Work

The future work will now concentrate on extensions to the presented architecture to allow creation of more advanced animation sequences. The extensions will then be implemented and evaluated, which will allow the avatars to perform more complex and varied activities. Next a scripting language will be investigated and added to the system to allow easier control of the animation parameters and modi-fication of settings governing the level of character interaction. A formalisation of the presented ideas may also be attempted.

## References

[1] Bauer B., Müller J. P., Odell J., Agent UML: A Formalism for Specifying Multiagent Software Systems, International Journal of Software Engineering and Knowledge En-gineering, Vol. 11, No. 3, pp1-24, 2001

[2] Blumberg B., Downie M., Ivanov Y., Berlin M., Johnson M. P., Tomlinson B., Inte-grated learning for interactive synthetic characters, in Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques, San Antonio, Texas, ACM Press, pp. 417 - 426, 2002

[3] Flake S., Geiger C., Küster J. M., Towards UML-based Analysis and Design of Multi-Agent Systems, in Proceedings of International NAISO Symposium on Information Science Innovations in Engineering of Natural and Artificial Intelligent Systems (ENAIS'2001), Dubai, March 2001

[4] Forte P., Szarowicz A., The Application of AI Techniques for Automatic Generation of Crowd Scenes, The Eleventh International Symposium on Intelligent Information Systems, Advances in Soft Computing pp. 209-216, Physica-Verlag, Sopot, Poland, 2002

[5] Funge J. D., AI for Games and Animation. A Cognitive Modeling Approach, A K Pe-ters Natick, Massachusetts, 1999

[6] Isla D., Burke R., Downie M., Blumberg B., A Layered Brain Architecture for Synthetic Creatures, pp. 1051-1058, in Proceedings of Seventeenth Joint Conference on Artificial Conference IJCAI-01, 4-10 August, Seattle, USA, 2001

[7] Laird J. E., Newell A., Rosenbloom P. S., Soar: An architecture for general intelligence, Artificial Intelligence, vol. 33 (1), pp1-64, 1987

[8] Laird, J. E. and Jones R. M., Building advanced autonomous AI systems for large scale real time simulations, Computer Games Development Conference. Long Beach, CA, 1998

[9] Mylopoulos J., Kolp M., Castro J., UML for Agent-Oriented Software Development: The Tropos Proposal, in Proceedings of the Fourth International Conference on the Unified Modeling Language, Toronto, Canada, October 2001
[10] Newell A., Unified Theories of Cognition, Harvard Press, 1990
[11] Padgham L. and Winikoff M., Prometheus: A Methodology for Developing Intelligent Agents, In proceedings of the Third International Workshop on Agent-Oriented Soft-ware Engineering, at AAMAS'02.
[12] Rao A. S., Georgeff M. P., Modeling Rational Agents within a BDI-Architecture, Pro-ceedings of the 2nd International Conference on Principles of Knowledge Representa-tion and Reasoning pp 473–484, Cambridge, MA, USA, April 1991, Morgan Kauf-mann Publishers, 1991
[13] Rao A. S., Georgeff M. O., BDI Agents: From Theory to Practice, in Proceedings of the First International Conference on Multiagent Systems ICMAS95, June 12-14, 1995
[14] Szarowicz A., Amiguet Vercher J., Forte P., Briggs J. H., Gelepithis P., Remagnino P., The Application of AI to Automatically Generated Animation, 14th Australian Joint Conference on Artificial Intelligence AI01, LNAI 2256: Advances in Artificial Intelli-gence pp. 487-494, Springer-Verlag, Adelaide, Australia, 2001
[15] Winikoff M., Padgham L., and Harland J., Simplifying the Development of Intelligent Agents, in AI2001: Advances in Artificial Intelligence, 14th Australian Joint Confer-ence on Artificial Intelligence, LNAI 2256, pp 557-568, Adelaide, December 2001
[16] Wood M. and DeLoach S. A., An Overview of the Multiagent Systems Engineering Methodology, in Agent-Oriented Software Engineering. P. Ciancarini, M. Wooldridge, (Eds.) LNAI Vol. 1957, Springer Verlag, Berlin, January 2001
[17] Wooldridge M., Jennings N. R., David Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, Autonomous Agents and Multi-Agent Systems, Vol. 3, No. 3, pp285-312, 2000
[18] Wray, Soar: A Functional Approach to General Intelligence, accessed from: http://ai.eecs.umich.edu/ soar/docs.html, 2002
[19] Yoon S.-Y, Blumberg B. M., Schneider G. E., Motivation Driven Learning for Inter-active Synthetic Characters, in Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain, ACM Press pp. 365-372, 2000

# An Abductive Interpretation for Open Agent Societies

M. Alberti[1], M. Gavanelli[1], E. Lamma[1], P. Mello[2], and P. Torroni[2]

[1] Dipartimento di Ingegneria, Università degli Studi di Ferrara
Via Saragat, 1, 44100 Ferrara, Italy
{malberti,mgavanelli,elamma}@ing.unife.it
[2] DEIS, Università degli Studi di Bologna
Viale Risorgimento, 2, 40136 Bologna, Italy
{pmello,ptorroni}@deis.unibo.it

**Abstract.** The focus of this work is on the interactions among (possibly heterogeneous) agents that form an open society, and on the definition of a computational logic-based architecture for agent interaction. We propose a model where the society defines the allowed interaction protocols, which determine the "socially" allowed agent interaction patterns. The semantics of protocols can be defined by means of social integrity constraints. The main advantages of this approach are in the design of societies of agents, and in the possibility to detect undesirable behavior. In the paper, we present the model for societies ruled by protocols expressed as integrity constraints, and its declarative semantics. A sketch of the operational counterpart is also given.

## 1 Introduction

In the Multi-Agent Systems community, societies of agents and agent interactions have been widely studied, and several models have been proposed.

Agent Communication Languages (ACL) and Conversation Protocols (CP) are the traditional approaches to support interactions among software agents. The semantics of speech acts in ACL is customarily defined in terms of mental attitudes such as Beliefs, Desires and Intentions [1]. This approach has been criticized as inadequate for open environments [2] since agents cannot verify whether the private beliefs of other agents comply with speech act definitions without pre-established constraints on how agents are internally implemented. On the other hand, CP are static structures that define the sequences of utterances making a coherent conversation. This approach has been criticized for its lack of flexibility, i.e., the lack of compositional rules governing how protocols are extended or merged.

In this work, we define the semantics of protocols as integrity constraints over social events (e.g., communicative acts), which caters for heterogeneity and openness, since it makes no assumptions on the internal structure of the agents.

We call such integrity constraints "social integrity constraints" ($IC_S$). $IC_S$ express "constraints" on the communication patterns of agents, and therefore determine "expected" communicative acts, on the basis of the history of

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 287–299, 2003.
© Springer-Verlag Berlin Heidelberg 2003

social events. The semantics for the ACL can be given in a uniform way, as done in [3, 4]. The overall proposed model for society protocols and ACL semantics has been developed in the context of the IST-2001-32530 project of the IST programme of the European Commission, titled *A computational Logic Model for the Description, Analysis and Verification of Global and Open Societies of Heterogeneous Computees - SOCS* [5].

Building on previous work on abductive logic-based agents, we assimilate the society's knowledge to abductive logic programs: we define a notion of *expected* social events, and express them as abducible predicates, while using $IC_S$ to constrain the "socially admissible" communication patterns of agents. The syntax of $IC_S$ and of the society in general are those of an extended logic program, and the semantics is inspired to that of abductive proof procedures such as the IFF [6]. This could lead to different kinds of verification: static, dynamic, and based on outside observation of the agents communication exchanges.

## 2 Society and Protocols

An open agent society needs to be tolerant to partial information, by continuing to operate despite the incompleteness of the available information. Also, the model of such a society should cope with the fact that information on agent interactions becomes available bit by bit over time. In our model, the society is time by time aware of social events that dynamically happen in the social environment. Moreover, the society can generate a set of "expected social events" that are not yet available to it. This set reflects the "ideal" behavior of the society and its members. Such expectations can be used by the society to behave proactively: they could be made public in order to try and influence the behavior of the agents, towards an ideal behavior.

Indeed, the expectations of the society are adjusted when it acquires new knowledge from the environment on social events that were not previously available. In this perspective, the society should be able to deal with unexpected social events from the environment, which violate the previous expectations.

This can be the case in an open environment where "regimentation" [7] cannot be assumed. Unexpected events can raise mechanisms of recovery from violation (e.g., sanctions), without affecting the society.

The knowledge in a society is composed of three parts: *Social Organization Knowledge Base* (SOKB), *Social Environment Knowledge Base* (SEKB), and a set $\mathcal{IC}_S$ of Social Integrity Constraints. $\mathcal{IC}_S$ is composed of $IC_S$ expressing what is expected to happen or is expected not to happen, given some history of events. For example, an $IC_S$ could state that the manager of a resource should give an answer to whomever has made a request for that resource. $IC_S$ can produce expectations on the future.

The SOKB defines structure and properties of the society, namely: goals, roles, and common knowledge and capabilities. SOKB can change from time to time. However, this knowledge can be seen as *static* since it describes the

organization of a society which changes more slowly than the way the SEKB does.

The current instantiation of a society is described by the SEKB, which takes into account occurred events and expectations about social events. Thus, the SEKB dynamically evolves (it is updated much more frequently than the SOKB) and consists of:

- The set of happened events (history **HAP**), containing all the events in the form of atoms $\mathbf{H}(p)$ where $p$ is a (ground) term. It represents the effects of agents' actions at the social level. During the evolution of the society via communication and interaction amongst agents, the set **HAP** can only dynamically grow, when new socially significant events happen.
- A set of expected events **EXP**, containing *expectations* on the future: events that should (but might not) happen in the future (indicated as atoms $\mathbf{E}(X)$, where $X$ is a term), and events that should not (but might indeed) happen in the future (indicated as atoms $\mathbf{NE}(X)$).

Intuitively, an **H** atom represents a socially significant event that happened in the society, i.e., social events are mapped into **H** predicates.

**E** is a positive expectation about an event (the society expects the event to happen in order to fulfill its protocols) and **NE** is a negative expectation, (i.e., the society expects the event not to happen). Expectations can be seen as hypotheses of the society about the behavior of agents. Note the difference between $\neg\mathbf{E}(X)$ and $\mathbf{NE}(X)$. The first expresses the fact that the society does not have an expectation about the happening of event $X$ (yet, if the event happens, no protocol will be violated), while the second expresses the fact that the society expects the event not to happen. By default (i.e., unless specified by **E** or **NE**), the society does not have expectations about events.

### 2.1 Syntax of the Social Organization Knowledge Base

We consider the $SOKB$ as a logic program; its syntax, in Backus-Naur form, is:

$$\begin{aligned}
SOKB &::= [Clause]^\star \\
Clause &::= Atom \leftarrow Cond \\
Cond &::= ExtLiteral\ [\ \wedge ExtLiteral\ ]^\star \\
ExtLiteral &::= Literal \mid Expectation \mid Constraint \\
Expectation &::= [\neg]\mathbf{E}(Term\ [,T]) \mid [\neg]\mathbf{NE}(Term\ [,T]) \\
Literal &::= Atom \mid \neg Atom \mid true
\end{aligned} \tag{1}$$

*Atom* and *Term* are intended as usual in Logic Programming [8]; *Constraint* is a constraint in the CLP sense [9].

$T$ is an optional parameter representing the *time* in which the expectation holds. It is a variable with a finite domain that can be subject to constraints [9].

The variables are quantified as follows:

- Universally, if they appear only in literals of kind **NE** (and possibly constraints) with scope the singleton **NE**;
- Otherwise universally with scope the entire clause.

Both goal-directed and non goal-directed behavior for a society is supported. In our abductive understanding of the society expectation, a goal-directed society may expect a certain behavior from an agent in order to reach its goal. As an example, we can consider a society with the goal of selling items. In order to sell an item, the society might expect some "auctioneer" agent to open the auction. The goal of the society could be $\leftarrow sold(nail)$ and the society might have, in the SOKB, a rule of kind:

$$sold(Item) \leftarrow \mathbf{E}(tell(Auctioneer, Bidders, openauction(Item, Dialogue)), T)$$

which says that one way to sell an item is to have some agent telling a set of possible bidders that an auction is open for the item. The protocol of the auction (i.e., the way the auctioneer and the bidders interact) can be then specified in $\mathcal{IC}_S$.

### 2.2 Syntax of Social Integrity Constraints

The set $\mathcal{IC}_S$ relates socially significant (happened) events and expected events. Intuitively, $IC_S$ in $\mathcal{IC}_S$ are (forward) rules used to produce *expectations* about the behavior of agents. They are used to check if an agent of the society behaves in a permissible way with respect to its "social" behavior. Their syntax, in BNF, is:

$$\begin{aligned}
\mathcal{IC}_S &::= [IC_S]^\star \\
IC_S &::= Body \rightarrow Head \\
Body &::= (Event|Expectation)\ [\wedge BodyLiteral]^\star \\
BodyLiteral &::= Event|Expectation|Literal|Constraint \\
Head &::= HeadDisjunct\ [\ \vee HeadDisjunct\ ]^\star|\bot \\
HeadDisjunct &::= Expectation\ [\ \wedge (Expectation|Constraint)]^\star \\
Expectation &::= [\neg]\mathbf{E}(Term\ [,T]) \mid [\neg]\mathbf{NE}(Term\ [,T]) \\
Event &::= [\neg]\mathbf{H}(Term\ [,T]) \\
Literal &::= Atom \mid \neg Atom \mid true
\end{aligned} \tag{2}$$

The rules of scope and quantification are as follows:

1. A variable must appear at least in an *Event* or in an *Expectation*.
2. The variables that appear both in the *Body* and in the *Head* are quantified universally with scope the entire $IC_S$.
3. The variables that appear only in the *Head* must appear in at least one *Expectation* (in eq. 2), have as scope the disjunct they belong to, and
   (a) if they appear in literals **E** or $\neg$**E** are quantified existentially;
   (b) otherwise they are quantified universally.
4. The variables that appear only in the *Body* have the *Body* as scope and
   (a) if they appear only in conjunctions of $\neg$**H**, **NE**, $\neg$**NE** or *Constraints* are quantified universally;
   (b) otherwise are quantified existentially.
5. the quantifier $\forall$ has higher priority than $\exists$ (e.g., literals will be quantified $\exists\forall$ and not viceversa).

*Example 1.* If an agent $X$ says "ask" to an agent $Y$ during a conversation $D$, $Y$ is expected to answer back either "yes" or "no", but not both "yes" and "no" (for the sake of simplicity, we do not report the constraints on time variables):

$$\begin{array}{l}
\mathbf{H}(tell(X,Y,ask,D),T) \rightarrow \mathbf{E}(tell(Y,X,yes,D),T') \vee \ \mathbf{E}(tell(Y,X,no,D),T') \\
\mathbf{H}(tell(X,Y,yes,D),T) \rightarrow \mathbf{NE}(tell(X,Y,no,D),T') \\
\mathbf{H}(tell(X,Y,no,D),T) \ \ \rightarrow \mathbf{NE}(tell(X,Y,yes,D),T')
\end{array}$$

For this example, we make scope and quantification of variables explicit:

$$\begin{array}{l}
\forall X \forall Y \forall D(\exists T(\mathbf{H}(tell(X,Y,ask,D),T)) \rightarrow \exists T'(\mathbf{E}(tell(Y,X,yes,D),T')) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vee \ \exists T'(\mathbf{E}(tell(Y,X,no,D),T'))) \\
\forall X \forall Y \forall D(\exists T(\mathbf{H}(tell(X,Y,yes,D),T)) \rightarrow \forall T'(\mathbf{NE}(tell(X,Y,no,D),T'))) \\
\forall X \forall Y \forall D(\exists T(\mathbf{H}(tell(X,Y,no,D),T)) \ \ \rightarrow \forall T'(\mathbf{NE}(tell(X,Y,yes,D),T')))
\end{array}$$

An agent may fulfill its expected behavior or not, so expectations may be fulfilled (by a history) or not. In this example, $X$ telling "ask" to $Y$ generates an expectation which can be fulfilled if $Y$ answers back "yes" ($Y$ behaves "properly"). $Y$ violates the protocol, instead, if it says "no" after saying "yes", due to the second $IC_S$ ("improper" behavior of $Y$).

At all times there can be alternative sets of expectations, as the previous example has shown. The alternative sets can be computed by a suitable proof procedure and provided to the agents (pro-actively) as a range of possibilities to comply to protocols (Sect. 4).

## 3 Declarative Semantics

In the following, we introduce a series of successively more refined declarative semantics. The various declarative semantics offer a range of options for different proof procedures, and are a ground basis to identify relevant properties of the society and its protocols.

Through this section, we consider the ground version of society's knowledge base and integrity constraints, and we do not consider CLP constraints.

We first introduce the concept of *admissible set of social expectations*. Intuitively, given a society and a set of events **HAP**, an admissible set of social expectations consists of a set of expectations about social events that are compatible with the SOKB, the set **HAP**, and the set $\mathcal{IC}_S$.

**Definition 1.** *Given a society and a set of events* **HAP***, an* admissible set *of social expectations* **EXP** *is a set of expectations such that:*

$$SOKB \cup \mathbf{HAP} \cup \mathbf{EXP} \models \mathcal{IC}_S \tag{3}$$

Many different sets of expectations are admissible, given $\mathcal{IC}_S$, SOKB, and **HAP**.

*Example 2 (admissible set of expectations).* Consider the following situation:

- $SOKB = \emptyset$
- $\mathbf{HAP} = \{\mathbf{H}(tell(thomas, yves, start)), \mathbf{H}(tell(david, yves, stop))\}$
- $\mathcal{IC}_S = \{\mathbf{H}(tell(X, Y, start)) \rightarrow \mathbf{E}(pass(Y)),$
  $\mathbf{H}(tell(X, Y, stop)) \rightarrow \mathbf{NE}(pass(Y))\}$

$\mathbf{EXP}_1 = \{\mathbf{E}(pass(yves)), \mathbf{NE}(pass(yves))\}$ is an admissible set of expectation, w.r.t. the $SOKB$, $\mathbf{HAP}$, and $\mathcal{IC}_S$ above. Notice that any superset of $\mathbf{EXP}_1$ is also admissible. Instead, $\mathbf{EXP}_2 = \{\mathbf{E}(pass(yves))\}$ is not an admissible set of expectations, because $\mathbf{H}(tell(david, yves, stop)) \in \mathbf{HAP}$, $\mathbf{NE}(pass(yves)) \notin \mathbf{EXP}_2$, and thus the second integrity constraint in $\mathcal{IC}_S$ is violated.

Note that an admissible set of expectations could be self-contradictory (e.g., both $\mathbf{E}(p)$ and $\neg\mathbf{E}(p)$ may belong to an admissible set). More refined semantics can be given by identifying a subset of admissible expectation sets as intended semantics for a society. In particular, we are interested in those which are *coherent* and *consistent*:

**Definition 2.** *A set of social expectations* $\mathbf{EXP}$ *is* coherent *if and only if:*

$$\{\mathbf{E}(p), \mathbf{NE}(p)\} \not\subseteq \mathbf{EXP}$$

*Example 3 (coherent set of expectations).* Let us consider the situation presented in Example 2. $\mathbf{EXP}_2 = \{\mathbf{E}(pass(yves))\}$ is a coherent set of expectations (although it is not admissible w.r.t. the $SOKB$, $\mathbf{HAP}$, and $\mathcal{IC}_S$). On the other hand, $\mathbf{EXP}_1 = \{\mathbf{E}(pass(yves)), \mathbf{NE}(pass(yves))\}$ is not a coherent set of expectations (although it is admissible).

Trivially, we are not interested in sets of social expectations that, at the same time, require that a particular event $p$ should happen and should not happen.

**Definition 3.** *A set of social expectations* $\mathbf{EXP}$ *is* consistent *if and only if:*

$$\{\mathbf{E}(p), \neg\mathbf{E}(p)\} \not\subseteq \mathbf{EXP} \textit{ and } \{\mathbf{NE}(p), \neg\mathbf{NE}(p)\} \not\subseteq \mathbf{EXP}$$

*Example 4 (consistent set of expectations).* Modification of Example 2:

- $SOKB = \emptyset$
- $\mathbf{HAP} = \{\mathbf{H}(tell(thomas, yves, start)), \mathbf{H}(tell(david, yves, stop))\}$
- $\mathcal{IC}_S = \{\mathbf{H}(tell(X, Y, start)) \rightarrow \mathbf{E}(pass(Y)),$
  $\mathbf{H}(tell(X, Y, stop)) \rightarrow \neg\mathbf{E}(pass(Y))\}$

The intuitive meaning of the second constraint in $\mathcal{IC}_S$ is: *If I tell you "stop" then one should not expect that you pass.*

$\mathbf{EXP}_2 = \{\mathbf{E}(pass(yves))\}$ is a consistent set of expectations, although it is not admissible, since $\mathbf{H}(tell(david, yves, stop)) \rightarrow \neg\mathbf{E}(pass(yves)) \in \mathcal{IC}_S$, $\mathbf{H}(tell(david, yves, stop)) \in \mathbf{HAP}$, and $\neg\mathbf{E}(pass(yves)) \notin \mathbf{EXP}_2$. $\mathbf{EXP}_3 = \{\mathbf{E}(pass(yves)), \neg\mathbf{E}(pass(yves))\}$ is instead an admissible set of expectations, but it is not consistent.

We are not interested in sets of social expectations that are intrinsically inconsistent, i.e., that at the same time, expect something and do not expect the same thing. When no coherent (and consistent) admissible expectation set exists, and therefore an incoherence (or inconsistency) arises, it means that the society has been modelled in a wrong way.

We would like to stress that we do not assume that expected events actually happen. This is in accordance with an *open* view for society where social expectations are just a suggestion for what should be done (or not done). It can be the case that in a situation an expectation is raised, but the expected event does not happen (this might lead to a violation, and possibly a sanction).

A further refined semantics is then given by identifying, among coherent and consistent admissible expectation sets, those which are *fulfilled* by a set of events happened in a society. This reflects the *ideal* behavior of a society.

**Definition 4.** *Given a society and a set of events* **HAP***, a coherent and consistent admissible set of social expectations* **EXP** *is* fulfilled *if and only if:*

$$\mathbf{HAP} \cup \mathbf{EXP} \models \{\mathbf{E}(p) \rightarrow \mathbf{H}(p)\} \cup \{\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)\} \tag{4}$$

*Example 5 (fulfilled set of expectations).* Let us consider the following situation:

- $SOKB = \emptyset$
- $\mathbf{HAP}_1 = \{\mathbf{H}(tell(thomas, yves, start)), \mathbf{H}(pass(yves))\}$
- $\mathcal{IC}_S = \{\mathbf{H}(tell(X, Y, start)) \rightarrow \mathbf{E}(pass(Y))\}$

$\mathbf{EXP}_2 = \{\mathbf{E}(pass(yves))\}$ is a coherent, consistent and fulfilled admissible set of expectations, w.r.t $SOKB$, $\mathbf{HAP}_1$, and $\mathcal{IC}_S$. But if we consider a different history: $\mathbf{HAP}_2 = \{tell(thomas, yves, start)\}$, $\mathbf{EXP}_2$ is not a fulfilled expectations set w.r.t. $SOKB$, $\mathbf{HAP}_2$, and $\mathcal{IC}_S$ (still, it is admissible, coherent, and consistent).

Many different **EXP** sets are admissible with respect to social integrity constraints, the SOKB and the history **HAP**. By Definition 4, we select, among them, those where the happened events cover all the events that should happen, and none of the events that should not happen.

Notice that such a fulfilled **EXP** set might not exist, even if a coherent and consistent admissible expectation set exists. The reason is the *violation* of the protocol: some agent did not behave as expected, and some action will be taken to recover from this violation.

**Definition 5.** *Given a society and a set* **HAP** *of events, if each coherent and consistent admissible set of expectations is not fulfilled (i.e., if* $\mathbf{E}(p) \rightarrow \mathbf{H}(p)$ *or* $\mathbf{NE}(p) \rightarrow \neg\mathbf{H}(p)$ *are violated), then we say that* **HAP** *produces a* violation.

Until now, we did not deal with a possible *goal* of the society. If we want to consider a goal-directed society, then we introduce the following definition.

**Definition 6.** *Given a society, a goal $G$ and a set of events* **HAP**, *we say that $G$ is* achievable *iff there exists a coherent and consistent admissible set of social expectations* **EXP**, *such that:*

$$SOKB \cup \mathbf{HAP} \cup \mathbf{EXP} \models G \tag{5}$$

*Example 6 (achievable goal of a society).* Let us consider the following situation:

- $SOKB = \{G_1 \leftarrow \mathbf{E}(pass(yves)),\ G_2 \leftarrow \mathbf{E}(pass(david))\}$
- $\mathbf{HAP} = \{\mathbf{H}(tell(yves, david, stop))\}$
- $\mathcal{IC}_S = \{\mathbf{H}(tell(X, Y, start)) \rightarrow \mathbf{E}(pass(Y)),$
  $\mathbf{H}(tell(X, Y, stop)) \rightarrow \mathbf{NE}(pass(Y))\}$

$G_1$ is an achievable goal w.r.t. $SOKB$, **HAP**, and $\mathcal{IC}_S$, thanks to the coherent and consistent admissible set of social expectations $\mathbf{EXP} = \{\mathbf{E}(pass(yves))\}$.

On the other hand, there exists no coherent and consistent admissible set of social expectations to make $G_2$ achievable, given history **HAP**.

Note that the notion of goal achievability does not guarantee that the goal is really achieved, since expectations may not be fulfilled, i.e., the corresponding events that should happen can be possibly not generated, and vice-versa.

**Definition 7.** *Given a society, a goal $G$ and a set of events* **HAP**, *$G$ is* achieved *iff there exists a fulfilled coherent and consistent admissible set of social expectations* **EXP** *such that Eq. 5 holds.*

*Example 7 (achieved goal of a society).* Let us consider the following situation:

- $SOKB = \{G_1 \leftarrow \mathbf{E}(pass(yves)),\ G_2 \leftarrow \mathbf{E}(pass(thomas))\}$
- $\mathbf{HAP} = \{tell(yves, david, stop), pass(yves)\}$
- $\mathcal{IC}_S = \{\mathbf{H}(tell(X, Y, start)) \rightarrow \mathbf{E}(pass(Y)),$
  $\mathbf{H}(tell(X, Y, stop)) \rightarrow \mathbf{NE}(pass(Y))\}$

$G_1$ is an achieved goal. $G_2$ is achievable, but it has not yet been achieved.

## 4 The Society Knowledge as an Abductive Logic Program

In our approach, agents autonomously perform some form of reasoning, while the society management infrastructure is devoted to ensuring that the agents do not collide with established rules and protocols.

By specifying protocols as $IC_S$, we can exploit an associated proof procedure to be used directly by a Society Infrastructure for verification, by using the intensional knowledge on constraints. Allowed paths can be inferred and not explicitly stated, avoiding over-constrained protocols.

We represent the knowledge available at social level as an Abductive Logic Program (ALP) [10], since such knowledge is generally incomplete. The idea of using abduction to model agent interaction is derived from a work by Kowalski

and Sadri on abductive agents [11], where the abducibles are produced within an agent cycle, and represent actions in the external world.

Events, expectations and society knowledge and protocols can be smoothly recovered into an abductive framework, so to exploit well-assessed proof-theoretic techniques in order to check the compliance of the overall computation with respect to the expected social behavior.

Using abduction to record expectations allows for two features. First, it enlarges the dynamic knowledge available at social level during the agents' own reasoning, through knowledge acquisition. Second, it is a way to make relevant knowledge available to all the agents in the society.

At the society level, knowledge can therefore be represented as the triple: $\langle KB, \mathcal{E}, IC \rangle$ where:

- $KB$ is the knowledge base of the society. It includes the SOKB and happened events ($\mathbf{HAP} \subseteq SEKB$);
- $\mathcal{E}$ is a set of *abducible predicates*, standing for positive and negative expectations or their negation;
- $IC$ is the set of Social Integrity Constraints, $\mathcal{IC}_S$.

Abduction captures relevant events (or hypotheses about future events), and a suitably extended abductive proof-procedure can be used for integrity constraint checking. Given a goal $G$ at the society level (see also Section 3), then $G$ is achieved when[1]:

$$KB \vdash_{\mathbf{EXP}} G \tag{6}$$

and $\mathbf{EXP} \subseteq \mathcal{E}$ is a (coherent and consistent) set of abduced atoms such that

$$KB \cup \mathbf{EXP} \vdash IC \tag{7}$$

$$KB \cup \mathbf{EXP} \vdash \{\mathbf{E}(X) \rightarrow \mathbf{H}(X)\} \cup \{\mathbf{NE}(X) \rightarrow \neg\mathbf{H}(X)\} \tag{8}$$

if this last condition is not verified, then a *violation* occurs (see Def. 5).

Notice that Eq. 7 is the operational counterpart of Eq. 3 and Eq. 8 is that of Eq. 4. If the society is not goal-directed (no goal is specified for it), Eq. 6 is always satisfied for any set of expectations and, therefore, only equations 7 and 8 are significant (as in the declarative semantics, Section 3).

In this section we gave an abductive interpretation of the social framework. A suitable proof procedure will have to be defined in order to efficiently deal with such a semantics in a dynamic settig. In particular, the compliance verification to the specified $\mathcal{IC}_S$ and fulfillment check should be incremental (in order to detect violations as soon as possible), and the operational phases of equations from 6 to 8 should be interleaved properly. Complexity will also be an issue to address.

[1] We do not commit at this stage to any particular semantics for the $\vdash$ symbol. Many semantics indeed could be given, such as for instance the classical SLDNF derivation as usual in Logic Programming. $KB \nvdash G$ is a shorthand for $not(KB \vdash G)$. The symbol $\vdash_{\Delta}$ stands for an abductive derivation with set of abduced atoms $\Delta$.

The IFF proof procedure [6] or an extension of it could be a plausible candidate because (*i*) it employs integrity constraints with forward reasoning, like our $IC_S$, and (*ii*) it keeps updated a "frontier" of alternative derivations, that could be communicated to agents as a range of socially satisfactory behaviors.

## 5 Related Work

Considerable work has been devoted to studying the concepts of norms, commitment and social relations in the context of multi-agent systems [12], and to proposing architectures for developing agents with social awareness [13]. Our approach can be considered complementary to these efforts, since it is mainly focused on the definition of a society infrastructure based on computational logic to regulate and improve robustness of interaction in an open environment, where the internal architecture of the agents might be unknown.

We have advocated a *social* approach as in [2], where the semantics of agent interaction is defined in terms of the effects that it has at a social level. Following this approach, even if the agents mental state cannot be accessed, it is possible to verify whether communicating agents in a society comply to the protocols that regulate their interaction. A *protocol* specifies the "rules of encounter" governing a dialogue between agents. It will usually allow for several alternative utterances in every situation and the agent in question has to choose one according to its private *policy*. A good protocol will enable fruitful interaction in general. A good policy will benefit the agent using it. The protocol is *public*, while each agent's policy is *private*. Protocols are practically important because they may help to select the adequate answer to an incoming utterance, thus reducing the complexity of this task for an agent [14].

The idea of expected behavior can be considered related to *deontic logic* [15]; however, our claim is that we do not need the full power of the standard deontic logic, but only constraints on events that are expected to happen or not to happen. We do not use deontic operators, but instead we map them into predicates (**E** for positive and **NE** for negative expectations).

Our work is very close for the objective and methodology to the notable work on computational societies presented and developed in the context of the ALFEBIITE project [16], and the work by Singh [17] where a social semantics is exemplified by using a commitment-based approach. With this work we share the same view of an open society as that of [18].

Artikis et al. [18] present a theoretical framework for providing executable specifications of particular kinds of multi-agent systems, called open computational societies, and present a formal framework for specifying, animating and ultimately reasoning about and verifying the properties of open computational society: systems where the behavior of the members and their interactions cannot be predicted in advance. Differently from [18], we do not explicitly represent the institutional power of the members and the concept of valid action. Permitted are all social events that do not determine a violation, i.e., all events that are not explicitly forbidden are allowed, and this implements a sort of "open

world assumption" at a society level. Permission, when it needs to be explicitly expressed, is mapped into the negation of a negative expectation: $\neg\mathbf{NE}(\ldots)$.

The semantics of our model can be directly mapped in an abductive framework, where expectations can be confirmed (fulfilled) or disconfirmed (violated) by the history of the happened social events.

Sadri et al. [19] propose a framework for agent negotiation based on dialogue. The dialogue of agents is defined in a two-part setting as an ordered sequence of communication primitives. The generation of dialogues results from an abductive reasoning process taking place inside each agent during the *think* phase of its life cycle (the cycle being inspired by [11]). Our work shares the view of integrity constraints that provide new abducible atoms, but in our case the abducibles are *expectations* of the society about the future behavior of the agents, while in [19] they are used as communication primitives.

## 6 Conclusions

We have presented a computational-logic based approach for modelling interactions among (possibly heterogeneous) agents that form an open society. The allowed interaction protocols, which determine the "socially" allowed agent interaction patterns, are expressed as (social) integrity constraints. The main advantages of this approach are in the design of societies of agents, and in the possibility to detect undesirable behavior.

We have shown that the overall model can be mapped into an abductive logic programming framework, and presented a declarative semantics as clean extension of that for (extended) Logic Programming. This makes existing results for logic programming and monotonic reasoning re-usable in our context (see, for instance, [20]). Future work will be devoted to extend the IFF proof procedure [6], in order to make it applicable to our purposes.

We think that the proposed model addresses a basic aspect and a major engineering problem, paving the way for different kinds of verification: static, dynamic, and based on outside observation of the agents communication exchanges. Preliminary results on that have been presented in [21]. The full implementation of such verification is subject for future work.

The proposed model is independent from the agents' internals, and it is applicable to societies of heterogeneous agents. Nonetheless, if agents were aware of social expectations, they could plan and act appropriately in order to achieve them. Future work will be also devoted to studying the influence and integration of raised expectations within the agents' behavior cycle.

Finally, recovery from violation, i.e., deciding what to do when a violation occurs in a society, is also a fascinating future work subject.

## Acknowledgements

This work is partially funded by the Information Society Technologies programme of the European Commission under the IST-2001-32530 project.

## References


[1] Rao, A. S., Georgeff, M. P.: Modeling rational agents within a BDI-architecture. In Fikes, R., Sandewall, E., eds.: Proc. of KR&R-91, (Morgan Kaufmann) 473–484 287

[2] Singh, M.: Agent communication language: rethinking the principles. IEEE Computer (1998) 40–47 287, 296

[3] Alberti, M., Ciampolini, A., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Logic based semantics for agent communication languages. In: Proc. of the Workshop of Formal Approaches to Multi-Agent Systems (FAMAS03), Warsaw (2003) 288

[4] Alberti, M., Ciampolini, A., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: A social ACL semantics by deontic constraints. In V.Marik, J.Muller, M.Pechoucek, eds.: Proc. of the 3rd International/Central and Eastern European Conference on Multi-Agent Systems, Prague, Czech Republic (2003) To appear in LNCS. 288

[5] SOCS: A Computational Logic Model for the Description, Analysis and Verification of Global and Open Societies of Heterogeneous Computees. IST–2001–32530. Web page (2001) `http://lia.deis.unibo.it/research/socs`. 288

[6] Fung, T. H., Kowalski, R. A.: The IFF proof procedure for abductive logic programming. Journal of Logic Programming **33** (1997) 151–165 288, 296, 297

[7] Krogh, C.: Obligations in multiagent systems. In Proc. of the 5th Scandinavian Conference on Artificial Intelligence, Trondheim, Norway, ISO Press (1995) 19–30 288

[8] Lloyd, J. W.: Foundations of Logic Programming. Springer-Verlag (1987) 289

[9] Jaffar, J., Maher, M.: Constraint logic programming: a survey. Journal of Logic Programming **19-20** (1994) 503–582 289

[10] Eshghi, K., Kowalski, R. A.: Abduction compared with Negation by Failure. In: Proc. of the 6th International Conference on Logic Programming. (1989) 294

[11] Kowalski, R. A., Sadri, F.: From logic programming to multi-agent systems. Annals of Mathematics and Artificial Intelligence (1999) 295, 297

[12] Conte, R., Falcone, R., Sartor, G.: Special issue on agents and norms. Artificial Intelligence and Law **1** (1999) 296

[13] Castelfranchi, C., Dignum, F., Jonker, C., Treur, J.: Deliberative normative agents: Principles and architecture. In: ATAL'99. Number 1757 in LNCS 364–378 296

[14] Endriss, U., Maudet, N., Sadri, F., Toni, F.: Protocol conformance for logic-based agents. In: Proc. of IJCAI 2003, (Morgan Kaufmann Publishers) 296

[15] Wright, G.: Deontic logic. Mind **60** (1951) 1–15 296

[16] ALFEBIITE: A Logical Framework for Ethical Behaviour between Infohabitants in the Information Trading Economy of the universal information ecosystem. IST–1999–10298. Web page (1999) `http://www.iis.ee.ic.ac.uk/alfebiite/`. 296

[17] Yolum, P., Singh, M.: Flexible protocol specification and execution: applying event calculus planning using commitments. In Castelfranchi, Lewis Johnson, eds.: Proc. of AAMAS'02, (ACM) 527–534 296

[18] Artikis, A., Pitt, J., Sergot, M.: Animated specifications of computational societies. In Castelfranchi, Lewis Johnson, eds.: Proc. of AAMAS'02, (ACM) 1053–1061 296

[19] Sadri, F., Toni, F., Torroni, P.: An abductive logic programming architecture for negotiating agents. In Greco, Leone, eds.: Proc. JELIA'02. Vol. 2424 of LNCS. 419–431 297

[20] Brogi, A., Lamma, E., Mancarella, P., Mello, P.: A unifying view for logic programming with non-monotonic reasoning. TCS **184** (1997) 1–59 297

[21] Alberti, M., Gavanelli, M., Lamma, E., Mello, P., Torroni, P.: Specification and verification of agent interactions using social integrity constraints. In: Workshop on Logic and Communication in Multi-Agent Systems, Eindhoven, Netherlands (2003) To appear. 297

# Reasoning about Conversation Protocols in a Logic-Based Agent Language

Matteo Baldoni, Cristina Baroglio, Alberto Martelli, and Viviana Patti

Dipartimento di Informatica, Università degli Studi di Torino
C.so Svizzera 185, I-10149 Torino, Italy
{baldoni,baroglio,mrt,patti}@di.unito.it

**Abstract.** We present an approach to reasoning about conversations within the framework of a logic-based agent language. Our agent theory is based on a modal logic of actions and beliefs and permits the representation of communicative acts and conversation protocols, allowing agents to reason about them before their execution. The work is framed in a world wide web context, in which we show how reasoning about the interaction with web service providers can be exploited for personalizing the service fruition.

## 1 Introduction

In this work we face the problem of allowing a rational agent, which is situated in a multi-agent framework, to reason about the conversations that it could carry on with other agents as a means of pursuing some goal of its own interest.

The capability of reasoning about conversations is, indeed, extremely useful in different application domains. Besides the often mentioned robotic applications, where, for instance, groups of agents must trade how to cooperate in order to perform some task, also the web world offers many interesting scenarios. Let us consider, as an example, a software agent, which is a user personal assistant. The task that it performs is to search the web in order to find *web services*, according to the user's specification. The scientific literature offers some standard languages for describing web services (such as DAML-S [5] and WSDL [4]) and there already are systems for finding web services that perform tasks of interest (e.g. the UDDI framework, where the WSDL standard is exploited). Currently, these languages base the description of a service on the lists of the inputs and of the outputs respectively required and returned by it. Let us suppose that you asked the agent to "book two tickets at a cinema where they show Akira" and that it should "not give the credit card number". The agent should match the information that you supplied to the inputs and outputs of the available services and select one of them that satisfies the requests. If we take a closer look to the two conditions, however, we can see that while the first identifies the *kind* of service and the *goal* to achieve, the second condition constrains the way in which the *interaction* between the service provider and the personal assistant should be carried on. The personal assistant, however, has no way for knowing

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 300–311, 2003.
© Springer-Verlag Berlin Heidelberg 2003

how its interlocutor will carry on the interaction because the service description does not contain any behavioral information, as noticed by McIlraith et al. [3]. In our work we focus on a specific kind of behavioral information, the *interaction protocol* followed by the provider. Whether or not a provider is a software agent, for communicating with its clients, it will follow some possibly non-deterministic protocol, aimed at getting/supplying all the necessary information. The cinema booking service and our personal assistant can, then, be seen as the two actors of a conversation, determined by the protocol imposed by the provider. If the personal assistant could *reason about* the possible interactions outcoming from such a conversation protocol, it could either verify if the interaction may be *personalized* by following an execution path that satisfies all the user's requirements, or, when this is not possible, it could search for another provider.

We faced the problem of describing and reasoning about conversation protocols in an *agent logic programming* setting by extending the modal action and belief framework of the language DyLOG, introduced in [2], so to deal also with *communicative behaviors*. In particular, we model how an agent can reason about the interaction that it is going to enact with another agent, with the aim of proving if there is a possible execution of the communication protocol, after which a set of beliefs of interest (goal) will be true in the agent mental state. Such a form of reasoning implies making assumptions about the mental state of the *other* agent, the one ours wishes to interact with. We considered a conversation protocol as (possibly non-deterministic) procedure that specifies the complex communicative behavior of a *single* agent, based upon simpler FIPA-like communicative acts, where the agent can either play the part of the sender or of the receiver of a message.

## 2 The Agent Language

DyLOG, introduced in [2], is a logic programming language for specifying and reasoning about the behavior of rational agents. It is based on a modal logic for reasoning about *actions* and *beliefs* and permits to define both complex actions and sensing actions for information gathering. Agents programmed in DyLOG can choose a course of actions, conditioned by their beliefs about the world and about other agents, and can use sensors for acquiring knowledge. Originally, DyLOG did not encompass communicative acts, but since in a *multi-agent environment* conversations with other agents can be a means of pursuing some goal of interest, we aim at integrating in the language a *communication kit*, that includes both primitive speech acts and pre-defined conversation protocols, that the agents can use to communicate with one another. Our action logic accounts both for atomic and complex actions, or procedures. Atomic actions are either world actions, which affect the world, or mental actions, i.e. sensing or communicative actions which only affect the mental state of an agent by modifying its beliefs. The set of atomic action consists of the set $\mathcal{A}$ of the world actions, the set $\mathcal{C}$ of communicative acts, and the set $\mathcal{S}$ of sensing actions. For each atomic action $a$ and agent $ag_i$ we introduce the modalities $[a^{ag_i}]$ and $\langle a^{ag_i} \rangle$. $[a^{ag_i}]\alpha$ means that $\alpha$

holds after every execution of action $a$ by agent $ag_i$; $\langle a^{ag_i}\rangle\alpha$ means that there is a possible execution of $a$ (by $ag_i$) after which $\alpha$ holds. For each atomic action $a$ in $\mathcal{A}\cup\mathcal{C}$ we also introduce a modality $Done(a^{ag_i})$ for expressing that $a$ has been executed. $Done(a^{ag_i})\alpha$ is read "$a$ has been executed by $ag_i$; before its execution, $\alpha$ was true"[1]. The modality $\Box$ denotes those formulas that hold in all possible agent mental states. Our formalization of complex actions draws considerably from dynamic logic for the definition of action operators like sequence, test and non-deterministic choice. However, differently than [10], we refer to a *Prolog-like* paradigm: procedures are defined by means of (possibly recursive) Prolog-like clauses. For each procedure $p$, the language contains also the modalities $[p]$ and $\langle p\rangle$ (universal and existential modalities respectively).

The mental state of an agent is described in terms of a consistent set of *belief fluents*. We enriched the belief state of a DyLOG agent by allowing also nested beliefs, for representing what other agents believe and reasoning on how they can be affected by communicative actions. We use the modal operator $\mathcal{B}^{ag_i}$ to model the beliefs of agent $ag_i$. The modality $\mathcal{M}^{ag_i}$ is defined as the dual of $\mathcal{B}^{ag_i}$ ($\mathcal{M}^{ag_i}\varphi \equiv \neg\mathcal{B}^{ag_i}\neg\varphi$). Intuitively $\mathcal{M}^{ag_i}\varphi$ means that $ag_i$ consider $\varphi$ possible.

All the modalities of the language are normal, $\Box$ is reflexive and transitive; its interaction with action modalities is ruled by $\Box\varphi \supset [a^{ag_i}]\varphi$. The epistemic modality $\mathcal{B}^{ag_i}$ is serial, transitive and euclidean. The interaction of $Done(a^{ag_i})$ with other modalities is ruled by: $\varphi \supset [a^{ag_i}]Done(a^{ag_i})\varphi$ and $Done(a^{ag_j})\varphi \supset \mathcal{B}^{ag_i}Done(a^{ag_j})\varphi$ (awareness), with $ag_i = ag_j$ when $a^{ag_i} \notin \mathcal{C}$. A non-monotonic solution to the persistency problem is given, which consists in maximizing persistency assumptions about fluents after the execution of action sequences, in the context of an abductive characterization. Such a solution is a generalization of the one in [2] for dealing with persistency of *nested* beliefs [13].

**Beliefs.** Our agents are modelled as individuals, each having its *subjective* point of view; we do not model the real world, the only relevant characterization of the states concerning the internal dynamics of each agent: its *mental state*, modelled as a set of beliefs. A *belief state* contains what $ag_i$ (dis)believes about the world and about the other agents. Agents do not control *other* agents beliefs: when an agent $ag_j$ informs $ag_i$ about $l$, it cannot be sure that $ag_i$ will believe $l$; this will happen only if $ag_i$ believes $ag_j$ to be an authority about $l$. We denote by $l$ either a fluent literal ($f$ or $\neg f$), or a *done fluent*, i.e. a modal atom $Done(a^{ag_i})\top$ or its negation. The belief state includes both a (possibly negated) epistemic fluent $\mathcal{B}^{ag_i}l$ for each literal, and, for each agent $ag_j$, a (possibly negated) epistemic fluent for each of its beliefs ($\mathcal{B}^{ag_i}\mathcal{B}^{ag_j}l$); $ag_j$ may also be equal to $ag_i$. To guarantee the *consistency* of the belief state, we impose the modality $\mathcal{B}^{ag_i}$ to be *serial*. In the single agent case [2], seriality guarantees that it is not possible that both $\mathcal{B}^{ag_i}l$ and $\mathcal{B}^{ag_i}\neg l$ hold in the same state. In a multi-agent setting, when we deal with nested beliefs, seriality also guarantees that no agent ascribes inconsistent beliefs to other agents; so for each couple of agents $ag_i$, $ag_j$ and for each belief formula $\mathcal{B}^{ag_i}l$: neither it is possible that $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i}l$ and $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i}\neg l$ nor that $\mathcal{B}^{ag_j}\mathcal{B}^{ag_i}l$

[1] $Done(a^{ag_i})\top$ is read "the action $a$ has been executed by agent $ag_i$".

and $\mathcal{B}^{ag_j}\neg\mathcal{B}^{ag_i}l$ hold in the same state. From the seriality of the $\mathcal{B}^{ag_i}$ operators, the general formula schema for the rank 2 beliefs $\mathcal{B}^{ag_i}\mathcal{B}^{ag_j}\neg\varphi \supset \neg\mathcal{B}^{ag_i}\mathcal{B}^{ag_j}\varphi$ holds in our logic for any two agents $ag_i$ and $ag_j$. This property guarantees that when an inconsistency arises "locally" in the beliefs ascribed from $ag_i$ to other agents, then the beliefs of $ag_i$ itself will be inconsistent. In essence a belief state is a complete and consistent set of rank 1 and 2 epistemic fluents. It provides, for each agent $ag_i$, a three-valued interpretation of all the possible belief arguments, i.e. fluent literals, done fluents and epistemic fluents of rank 1: each belief argument $L$ is either *true*, *false*, or *undefined* when both $\neg\mathcal{B}^{ag_i}L$ and $\neg\mathcal{B}^{ag_i}\neg L$ hold. In the last case we use $\mathcal{U}^{ag_i}L$ for expressing the ignorance of $ag_i$ about $L$.

**Behavior.** In the line of [2] the *behavior* of an agent $ag_i$ can be specified by a domain description, which includes, besides a specification of the agent current beliefs: (1) action and precondition laws for describing the *atomic world actions* in terms of their preconditions and effects on the executor's mental state; (2) sensing axioms for describing *atomic sensing actions*; (3) procedure axioms for describing *complex behaviors*.

World actions are described by their preconditions and effects on the actor's mental state; they trigger a revision process on the actor's beliefs. Formally, *action laws* describe the conditional effects on $ag_i$'s belief state of an atomic action $a \in \mathcal{A}$, executed by $ag_i$ itself. They have the form:

$$\Box(\mathcal{B}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i}L_n \supset [a^{ag_i}]\mathcal{B}^{ag_i}L_0) \tag{1}$$

$$\Box(\mathcal{M}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{M}^{ag_i}L_n \supset [a^{ag_i}]\mathcal{M}^{ag_i}L_0) \tag{2}$$

Law (1) states that if $ag_i$ believes the preconditions to an action $a$ in a certain epistemic state, after $a$ execution, $ag_i$ will also believe the action's effects. (2) states that when the preconditions of $a$ are unknown to $ag_i$, after the execution of $a$, $ag_i$ will consider unknown also its effects[2].

*Precondition laws* specify mental conditions that make an action in $\mathcal{A} \cup \mathcal{C}$ executable in a state. They have form:

$$\Box(\mathcal{B}^{ag_i}L_1 \wedge \ldots \wedge \mathcal{B}^{ag_i}L_n \supset \langle a^{ag_i}\rangle\top) \tag{3}$$

The meaning is trivial: $ag_i$ can execute $a$ when the precondition fluents of $a$ are in $ag_i$'s belief state.

Sensing Actions produce knowledge about fluents; they are defined as non-deterministic actions, with unpredictable outcome, formally modelled by a set of *sensing axioms*. If we associate to each sensing action $s$ a set $dom(s)$ of literals (domain), when $ag_i$ executes $s$, it will know which of such literals is true:

$$[s]\varphi \equiv [\bigcup_{l \in dom(s)} s^{\mathcal{B}^{ag_i}l}]\varphi \tag{4}$$

$\cup$ is the choice operator of dynamic logic and $s^{\mathcal{B}^{ag_i}l}$, for each $l \in dom(s)$, is an *ad hoc* primitive action, that probes one of the possible outcomes of the sensing.

[2] Laws of form (2) allow actions with non-deterministic effects, that may cause a *loss* of knowledge, to be specified.

Notice that the binary sensing action on a literal $l$ is a special case of sensing where the associated finite set is $\{l, \neg l\}$.
Complex actions We specify agent complex behaviors by means of *procedure definitions*, built upon other actions. Formally, a complex action is defined by means of a collection of *inclusion axiom schema* of our modal logic, of form:

$$\langle p_0 \rangle \varphi \subset \langle p_1; p_2; \ldots; p_m \rangle \varphi \tag{5}$$

$p_0$ is a procedure name and the $p_i$'s ($i = 1, \ldots, m$) are either procedure names, atomic actions, or test actions; the operator ";" is the sequencing operator of dynamic logic. Procedure definitions may be recursive and procedure clauses can be executed in a goal directed way, similarly to standard logic programs.

## 3 Communication

The integration of a *communication theory* in the general agent theory came straightforwardly by adding further axioms and laws to $ag_i$'s domain description. In this section we enrich the specification of an agent behavior with a *communication kit*, that includes a predefined set of *primitive speech acts*, a set of special *get message actions*, and a set of *conversation protocols*.

Speech Acts Communication primitives are treated as atomic actions and described in terms of preconditions and effects on the agent mental state. Each communicative primitive action has the form $\mathsf{speech_act}(sender, receiver, l)$, where *sender* and *receiver* are agents and $l$ is the propositional content of the speech act[3]. As a difference with world actions, in the case of communication primitives, the sender aims at modifying the mental state of the receiver, possibly inducing a cooperative behavior. Such actions can be seen as special mental actions, affecting both the sender's and the receiver's mental state. In our model we focused on the *internal representation*, that each agent has of each speech act, by specifying $ag_i$'s belief changes both when $ag_i$ is the sender and when it is the receiver. Such a representation is necessary for providing the capability of *reasoning about* conversation effects. Hence, speech acts are modelled by generalizing the action and precondition laws of form (1), (2), and (3), so to allow the representation of the effects of communications performed by other agents on our agent mental state. Then, our speech act specification is twofold: one definition holds when the agent is the executor (the sender), the other when it is the receiver. In the first case, the precondition laws contain some *sincerity condition* that must hold in the agent mental state. When it is the receiver, the action is considered *always* executable. Let us define the primitive speech acts we will use, inform, queryIf, and refuseInform; a wider set can be found in [13].

[3] For the sake of simplicity we allow as propositional contents only attitude-free fluents, i.e fluent literals or done fluents.

$\mathsf{inform}(sender, receiver, l)$
a) $\Box(\mathcal{B}^{ag_i} l \wedge \mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l \supset \langle \mathsf{inform}(ag_i, ag_j, l)\rangle \top)$
b) $\Box([\mathsf{inform}(ag_i, ag_j, l)]\mathcal{M}^{ag_i}\mathcal{B}^{ag_j} l)$
c) $\Box(\mathcal{B}^{ag_i}\mathcal{B}^{ag_j} authority(ag_i, l) \supset [\mathsf{inform}(ag_i, ag_j, l)]\mathcal{B}^{ag_i}\mathcal{B}^{ag_j} l)$
d) $\Box(\top \supset \langle \mathsf{inform}(ag_j, ag_i, l)\rangle \top)$
e) $\Box([\mathsf{inform}(ag_j, ag_i, l)]\mathcal{B}^{ag_i}\mathcal{B}^{ag_j} l)$
f) $\Box(\mathcal{B}^{ag_i} authority(ag_j, l) \supset [\mathsf{inform}(ag_j, ag_i, l)]\mathcal{B}^{ag_i} l)$
g) $\Box(\mathcal{M}^{ag_i} authority(ag_j, l) \supset [\mathsf{inform}(ag_j, ag_i, l)]\mathcal{M}^{ag_i} l)$

Clause (a) states that an inform act can be executed when the sender believes $l$ and believes that the receiver does not know $l$. When $ag_i$ is the sender it thinks possible that the receiver will adopt its belief, although it cannot be certain (autonomy assumption (b)). If it believes to be a trusted *authority* about $l$, it is confident that the receiver will adopt its belief, (c). When $ag_i$ is the receiver of an inform act, it will believe that $l$ is believed by the sender $ag_j$ (e), but it will adopt $l$ as itw own belief only if it recognizes $ag_j$ as a trusted authority, (f)-(g).

$\mathsf{queryIf}(sender, receiver, l)$
a) $\Box(\mathcal{U}^{ag_i} l \wedge \neg\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l \supset \langle \mathsf{queryIf}(ag_i, ag_j, l)\rangle \top)$
b) $\Box(\top \supset \langle \mathsf{queryIf}(ag_j, ag_i, l)\rangle \top)$
c) $\Box([\mathsf{queryIf}(ag_j, ag_i, l)]\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l)$

By queryIf $ag_i$ asks $ag_j$ if it believes that $l$ is true. To perform a queryIf act, $ag_i$ must ignore $l$ and it must believe that the receiver does not ignore $l$ (a). After a queryIf act, the receiver will believe that the sender ignores $l$.

$\mathsf{refuseInform}(sender, receiver, l)$
a) $\Box(\mathcal{U}^{ag_i} l \wedge \mathcal{B}^{ag_i} Done(\mathsf{queryIf}(ag_j, ag_i, l))\top \supset \langle \mathsf{refuseInform}(ag_i, ag_j, l)\rangle \top)$
b) $\Box(\top \supset \langle \mathsf{refuseInform}(ag_j, ag_i, l)\rangle \top)$
c) $\Box([\mathsf{refuseInform}(ag_j, ag_i, l)]\mathcal{B}^{ag_i}\mathcal{U}^{ag_j} l)$

By refuseInform an agent refuses to give an information it was asked for. The refusal can be executed only if: the sender is ignorant about $l$ and it believes that the receiver previously queried it about $l$. After a refusal the receiver believes the sender to be ignorant about $l$.

Get Message Actions are used for *receiving* messages from other agents; they can be viewed as queries for an external input, whose outcome is unpredictable. We model them as a special kind of sensing actions, the main difference being that they are defined by means of speech acts. Formally, we represent the fact that $ag_i$ expects a communication from $ag_j$, by means of a get_message action, given by an axiom schema of form:

$$[\mathsf{get_message}(ag_i, ag_j, l)]\varphi \equiv [\bigcup_{\mathsf{speech_act} \in \mathcal{C}_{\mathsf{get_message}}} \mathsf{speech_act}(ag_j, ag_i, l)]\varphi \qquad (6)$$

having as defining primitive actions a finite set of speech acts $\mathcal{C}_{\mathsf{get_message}} \subseteq \mathcal{C}$. $\mathcal{C}_{\mathsf{get_message}}$ is the set of the possible communications that $ag_i$ expects from $ag_j$ in

the context of a given conversation. Then, we do not associate to a get_message action a domain of mental fluents, but we calculate the information obtained by looking at the effects of the speech acts in $\mathcal{C}_{\mathsf{get_message}}$ on $ag_i$'s mental state.

Conversation protocols Often primitive communication acts occur in the context of predefined conversation schemas [11] that specify communication patterns. To cope with these aspects we provide agents with conversation protocols. Each agent has a subjective perception of on-going communications, for this reason protocols have as many procedural representations as possible roles in the conversation. Let us consider, for instance the yes_no_query protocol reported hereafter, a simplified version of the FIPA Query Interaction Protocol [8]. The protocol has two representations, one to be followed for making a query ($\mathsf{yes_no_query}_Q$) and a complementary one for responding ($\mathsf{yes_no_query}_I$):

$$\langle \mathsf{yes_no_query}_Q(Self, Other, Fluent)\rangle\varphi \subset \\ \langle \mathsf{queryIf}(Self, Other, Fluent); \mathsf{get_answer}(Self, Other, Fluent)\rangle\varphi$$

$$[\mathsf{get_answer}(Self, Other, Fluent)]\varphi \equiv \\ [\mathsf{inform}(Other, Self, Fluent) \cup \mathsf{inform}(Other, Self, \neg Fluent) \cup \\ \mathsf{refuseInform}(Other, Self, Fluent)]\varphi$$

Intuitively, the right hand side of get_answer represents all the possible answers expected by agent $Self$ from agent $Other$ about $Fluent$, in the context of a conversation ruled by the $\mathsf{yes_no_query}_Q$ protocol.

$$\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\ \langle \mathsf{get_start}(Self, Other, Fluent); \\ \mathcal{B}^{Self} Fluent?; \mathsf{inform}(Self, Other, Fluent)\rangle\varphi$$

$$\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\ \langle \mathsf{get_start}(Self, Other, Fluent); \\ \mathcal{B}^{Self} \neg Fluent?; \mathsf{inform}(Self, Other, \neg Fluent)\rangle\varphi$$

$$\langle \mathsf{yes_no_query}_I(Self, Other, Fluent)\rangle\varphi \subset \\ \langle \mathsf{get_start}(Self, Other, Fluent); \\ \mathcal{U}^{Self} Fluent?; \mathsf{refuseInform}(Self, Other, Fluent)\rangle\varphi$$

Intuitively the $\mathsf{yes_no_query}_I$ protocol specifies the behavior of the agent $Self$, that waits a query from $Other$; afterwards, it replies according to its beliefs on the query subject. get_start is a get_message action ruled by the following axiom:

$$[\mathsf{get_start}(Self, Other, Fluent)]\varphi \equiv [\mathsf{queryIf}(Other, Self, Fluent)]\varphi$$

We define the *communication kit* of an agent $ag_i$, $\mathsf{CKit}^{ag_i}$, as the triple $(\Pi_{\mathcal{C}}, \Pi_{\mathcal{CP}}, \Pi_{\mathcal{S}get})$, where $\Pi_{\mathcal{C}}$ is the set of simple action laws defining $ag_i$'s primitive communicative speech acts, $\Pi_{\mathcal{S}get}$ is a set of axioms for $ag_i$'s get_message actions and $\Pi_{\mathcal{CP}}$ is the set of procedure axioms specifying $ag_i$'s conversation protocols. In this extension of the DyLOG language, the *Domain Description* for agent $ag_i$ is a triple $(\Pi, \mathsf{CKit}^{ag_i}, S_0)$, where $\mathsf{CKit}^{ag_i}$ is $ag_i$ communication kit, $S_0$ is the initial set of $ag_i$'s belief fluents, and $\Pi$ is a tuple $(\Pi_{\mathcal{A}}, \Pi_{\mathcal{S}}, \Pi_{\mathcal{P}})$, where $\Pi_{\mathcal{A}}$ is the set of $ag_i$'s action and precondition laws for world actions, $\Pi_{\mathcal{S}}$ is a set of axioms for $ag_i$'s sensing actions, $\Pi_{\mathcal{P}}$ a set of axioms that define complex actions.

## 4 Reasoning about Conversations

Given a domain description, we can reason about it and formalize the *temporal projection* problem and the *planning* problem, by means of existential queries having the form:

$$\langle p_1 \rangle \langle p_2 \rangle \ldots \langle p_m \rangle Fs \tag{7}$$

Each $p_k$, $k = 1, \ldots, m$ in (7) may either be an (atomic or complex) action executed by $ag_i$ or an external speech act, that belongs to $\mathsf{CKit}^{ag_i}$ (by the word *external* we denote a speech act in which our agent plays the role of the receiver). By checking if a query of form (7) succeeds we can cope with the planning problem. In fact this corresponds to answering the question "is there an execution trace of $p_1$, ..., $p_n$ leading to a state where the conjunction of belief fluents $Fs$ holds for $ag_i$?". Such an execution trace is a plan to bring about $Fs$. The procedure definition constrains the search space. Notice that when all the $p_k$ in the query are atomic actions that belong to $\mathcal{A} \cup \mathcal{C}$, by checking if the query succeeds, we cope also with the temporal projection problem: "does $Fs$ hold for $ag_i$, after the execution of the action sequence $a_1, \ldots, a_m$?".

In presence of communication, the planning and the temporal projection problems turn respectively into the problem of reasoning about *conversation protocols* and reasoning about simple *conversations*, where a conversation is a sequence of speech acts. This allows, for instance, an agent to *investigate* the possible changes to its mental state, produced by a specific conversation, or if a conversation is an instance of some predefined protocol [7].

In the case of planning, the action sequence will contain both actions in which the agent is the sender and actions in which it is the receiver. In this process we treat get_message actions as sensing actions, whose outcome cannot be known at planning time. Since agents cannot read each other's mind, they cannot know in advance the answers that they will receive. For this reason all of the possible alternatives are to be taken into account (and we can foresee them because of the existence of the protocol). Therefore, the extracted plan will be *conditional*, meaning that for each get_message and for each sensing action, it will contain as many branches as possible action outcomes. Each path in the resulting tree is a linear plan that brings about the desired condition $Fs$. More formally, a conditional plan $\sigma$ is either:

- a action sequence $a_1; \ldots; a_m$, with $m \geq 0$;
- if $a_1; \ldots; a_m$ is an action sequence ($m \geq 0$), $s \in \mathcal{S}$ is a sensing action, and $\sigma_1, \ldots, \sigma_t$ are conditional plans then $a_1; \ldots; a_m; s; ((\mathcal{B}^{ag_i} l_1?); \sigma_1 \cup \ldots \cup (\mathcal{B}^{ag_i} l_t?); \sigma_t)$, where $l_1, \ldots, l_t \in dom(s)$;
- if $a_1; \ldots; a_m$ is an action sequence ($m \geq 0$), $g \in \mathcal{S}$ is a get_message action, and $\sigma_1, \ldots, \sigma_t$ are conditional plans then $a_1; \ldots; a_k; g; ((\mathcal{B}^{ag_i} Done(c_1) \top?); \sigma_1 \cup \ldots \cup (\mathcal{B}^{ag_i} Done(c_t) \top?); \sigma_t)$, where $c_1, \ldots, c_t \in \mathcal{C}_g$.

The proof procedure is a natural evolution of the work in [2], and is described in [1]; it is goal-directed and based on negation as failure (NAF). NAF is used

to deal with the persistency problem to verify that the complement of a mental fluent is not true in the state resulting from an action execution; while in the modal theory we adopted an abductive characterization [13]. The proof procedure allows agents to find *linear* and *conditional* plans for achieving a goal from an incompletely specified initial state. The soundness can be proved under the assumption of e-consistency, i.e. for any action the set of its effects is consistent [6]. Moreover, the extracted plans always lead to a state in which the desired condition *Fs* holds, for all the possible results of the sensing actions.

## 5 An Example in the Web Services Scenario

Let us consider the personal assistant (*pa*) mentioned in the introduction, its aim is to search a *cinema booking service* that satisfies the user's requests. Suppose that two services are available, *click_ticket* and *all_cinema*, that they follow two different interaction protocols (get_ticket_1 and get_ticket_2), see Fig. 1, and that such protocols are part of the web service descriptions. The difference between them is that the former permits both to book a ticket to be paid later by cash and to buy it by credit card, while the latter allows only ticket purchase by credit card. Let us consider the DyLOG specifications of the two protocols, that describe the interaction of *pa* with the web services from the *customer perspective*: $\mathsf{get_ticket_1}_C$ and $\mathsf{get_ticket_2}_C$. We will suppose that *pa* knows the credit card number (*cc_number*) of the user but that it is requested not to use it. Let us see how *pa* reasons on the *way* in which conversations will be carried on.

(a) $\langle \mathsf{get_ticket_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes_no_query}_Q(Self, WebS, available(Film));$
$\mathcal{B}^{Self} available(Film)?\ ;\ \mathsf{get_info}(Self, WebS, cinema(C));$
$\mathsf{yes_no_query}_I(Self, WebS, pay_by(credit_card));$
$\mathcal{B}^{Self} pay_by(credit_card)?\ ;\ \mathsf{inform}(Self, WebS, cc_number);$
$\mathsf{get_info}(Self, WebS, booked(Film))\rangle\varphi$

(b) $\langle \mathsf{get_ticket_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes_no_query}_Q(Self, WebS, available(Film));$
$\mathcal{B}^{Self} available(Film)?\ ;\ \mathsf{get_info}(Self, WebS, cinema(C));$
$\mathsf{yes_no_query}_I(Self, WebS, pay_by(credit_card));$
$\neg\mathcal{B}^{Self} pay_by(credit_card)?\ ;\ \mathsf{get_info}(Self, WebS, pay_by(cash));$
$\mathsf{get_info}(Self, WebS, booked(Film))\rangle\varphi$

(c) $\langle \mathsf{get_ticket_1}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes_no_query}_Q(Self, WebS, available(Film)); \neg\mathcal{B}^{Self} available(Film)?\rangle\varphi$

(d) $[\mathsf{get_info}(Self, WebS, Fluent)]\varphi \subset [\mathsf{inform}(WebS, Self, Fluent)]\varphi$

Protocol $\mathsf{get_ticket_1}_C$ works in the following way: the personal assistant ($Self$) is supposed to begin the interaction. After checking if the requested movie is available in some cinema by the $\mathsf{yes_no_query}_Q$ protocol, it waits for an information (get_info) from the provider ($WebS$) about which cinema shows it. Then the form of payment is defined: (a) encodes the interaction occuring when the tickets are paid by credit card (see Fig. 1); (b) is selected when $\neg\mathcal{B}^{Self} pay_by(credit_card)$

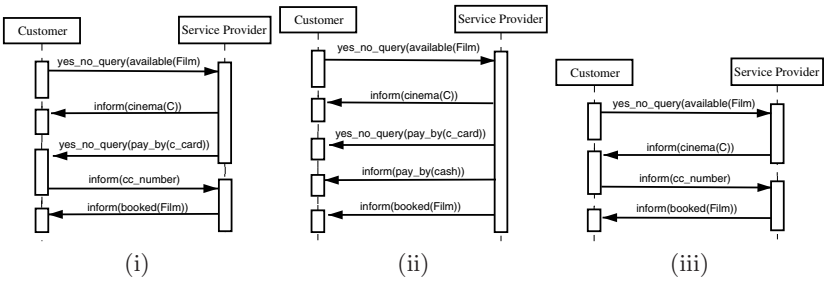



**Fig. 1.** The three AUML graphs [12] represent the communicative interactions occurring between the customer ($pa$) and the provider.

is contained in $pa$ mental state (our case), leading to book a ticket to be paid cash. In both cases a confirmation of the ticket booking is returned to $pa$. Clause (c) tackles the case in which the movie is not available; (d) describes get_info, which is a get_message action. Let us now consider the query:

$$\langle \mathsf{get_ticket_1}_C(pa, click_ticket, akira)\rangle \mathcal{B}^{pa} \neg \mathcal{B}^{click_ticket} cc_number$$

that amounts to determine if there is a conversation between $pa$ and $click_ticket$ about the movie $akira$, that is an instance of protocol $\mathsf{get_ticket_1}_C$, after which the service does not know the credit card number of the user. Agent $pa$ works on the behalf of a user, thus it knows his credit card number ($B^{pa} cc_number$) and his desire not to use it in the current transaction ($\neg \mathcal{B}^{pa} pay_by(credit_card)$). It also believes to be an authority about the form of payment and about the user's credit card number and that $click_ticket$ is an authority about cinema and tickets. This is represented by the beliefs: $\mathcal{B}^{pa} authority(pa, cc_number)$ and $\mathcal{B}^{pa} authority(click_ticket, booked(akira))$. The initial mental state will also contain the fact that $pa$ believes that no ticket for $akira$ has been booked yet, $\mathcal{B}^{pa} \neg booked(akira)$, and some hypothesis on $click_ticket$ mental state, e.g. the belief fluent $\mathcal{B}^{pa} \neg B^{click_ticket} cc_number$, meaning that the web service does not already know the credit card number. Suppose, now, that the ticket is available; since $pa$ mental state contains the belief $\neg \mathcal{B}^{pa} pay_by(credit_card)$, when it reasons about the protocol execution, the test on $\mathcal{B}^{pa} pay_by(credit_card)?$ fails. Then clause (b) is to be followed, leading $pa$ to be informed that it booked a ticket, $\mathcal{B}^{pa} booked(akira)$, which is supposed to be paid cash. No communication involves the belief $\mathcal{B}^{pa} \neg B^{click_ticket} cc_number$, which persists from the initial state. Even when the ticket is not available or the movie is not known by the provider, the interaction ends without consequences on such a fluent. After the briefly described reasoning process, the agent finds an execution trace of $\mathsf{get_ticket_1}_C$, which corresponds to a *personalized conditional dialogue plan* between itself and the provider $click_ticket$, always leading to satisfy the user goal of not giving the credit card number:

$\mathsf{queryIf}(pa, click_ticket, akira);$
$\quad ((\mathcal{B}^{pa} Done(\mathsf{inform}(click_ticket, pa, akira))\top?);$
$\quad\quad \mathsf{get_info}(pa, click_ticket, cinema(C));$
$\quad\quad (\mathcal{B}^{pa} Done(\mathsf{inform}(click_ticket, pa, cinema(C)))\top?);$

$\mathsf{get_info}(pa, click_ticket, pay_by(credit_card)));$
$(\mathcal{B}^{pa}\, Done(\mathsf{queryIf}(click_ticket, pa, pay_by(credit_card)))\top?);$
$\mathsf{inform}(pa, click_ticket, \neg pay_by(credit_card));$
$\mathsf{get_info}(pa, click_ticket, pay_by(cash));$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click_ticket, pa, pay_by_cash))\top?);$
$\mathsf{get_info}(pa, click_cinema, booked(akira));$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click_ticket, pa, booked(akira)))\top?) \cup$
$(\mathcal{B}^{pa}\, Done(\mathsf{inform}(click_ticket, pa, \neg akira))\top?) \cup$
$(\mathcal{B}^{pa}\, Done(\mathsf{refuseInform}(click_ticket, pa, akira))\top?))$

Let us now considering $\mathsf{get_ticket_2}_C$, the protocol followed by $all_cinema$:

(e) $\langle \mathsf{get_ticket_2}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes_no_query}_Q(Self, WebS, available(Film));$
$\mathcal{B}^{Self} available(Film)?\ ;\ \mathsf{get_info}(Self, WebS, cinema(C));$
$\mathsf{inform}(Self, WebS, cc_number);$
$\mathsf{get_info}(Self, WebS, booked(cinema_ticket, Film))\rangle\varphi$

(f) $\langle \mathsf{get_ticket_2}_C(Self, WebS, Film)\rangle\varphi \subset$
$\langle \mathsf{yes_no_query}_Q(Self, WebS, available(Film)); \neg\mathcal{B}^{Self} available(Film)?\rangle\varphi$

We can easily see that in this case, the query $\langle \mathsf{get_ticket_2}_C(pa,\ all_cinema,\ akira)\rangle\ \mathcal{B}^{pa}\neg\mathcal{B}^{all_cinema} cc_number$ fails because, the protocol allows only one interaction sequence, containing the action $\mathsf{inform}(pa,\ all_cinema,\ cc_number)$ that causes the mental state of $pa$ to contain $\mathcal{B}^{pa}\mathcal{B}^{all_cinema}\ cc_number$.

## 6 Conclusions

We have presented an approach to reasoning about conversation protocols within the framework of a logic-based agent language. We have studied how to embed a theory of communicative actions in the DyLOG logical framework [2] so to allow the modelling of software agents that, when situated in a multi-agent environment, can interact with one another by a speech act based communication mechanism. In the extended language agents have their own local beliefs on the world and on the other agents mental state. The semantics of communicative acts is described in terms of their effects on the mental state of both the sender and the recipient. In the line of [11], we used conversation protocols to provide our agents decision procedures for suitably responding to communications. We took a subjective representation of conversation protocols, by making hypothetical assumptions on the other's answers. As a consequence protocols have been easily integrated with other policies defining the agent behavior, being both represented as procedures specified in DyLOG. Notice that, since we are only interested in reasoning about the local mental state dynamics, our approach differs from other logic-based approaches to communication in multi-agent systems, as the ones taken in [15, 9], where communicative acts affect the global state of a multi-agent system and the target is to verify global properties of the overall multi-agent system execution. Instead, our focus on the internal specification of interaction protocols for planning dialogue moves is closer to the one taken in [14], where

negotiation protocols, expressed by sets of dialogue constraints, are included in the agent program and used for triggering dialogues that achieve goals. However such an approach leads to different results and, being somehow focused on goal achievement in a multi-agent environment with limited resources, is not aimed at implementing the reasoning about conversations we focused on.

## References


[1] M. Baldoni, C. Baroglio, A. Martelli, and V. Patti. Reasoning about self and others: communicating agents in a modal action logic. In R. Gorrieri, C. Blundo, and C. Laneve, editors, *Proc. of ICTCS'2003*, LNCS, Bologna, Italy, 2003. Springer. To appear. 307

[2] M. Baldoni, L. Giordano, A. Martelli, and V. Patti. Reasoning about Complex Actions with Incomplete Knowledge: A Modal Approach. In *Proc. of ICTCS'2001*, volume 2202 of *LNCS*, pages 405–425. Springer, 2001. 301, 302, 303, 307, 310

[3] J. Bryson, D. Martin, S. McIlraith, and L. A. Stein. Agent-based composite services in DAML-S: The behavior-oriented design of an intelligent semantic web, 2002. 301

[4] R. Chinnici, M. Gudgin, J. J. Moreau, and S. Weerawarana. Web Services Description Language (WSDL) version 1.2, 2003. Working Draft. 300

[5] The DAML-S coalition. DAML-S: Web service description for the semantic web. In *the 1st Int. Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002. 300

[6] M. Denecker and D. De Schreye. Representing Incomplete Knowledge in Abduction Logic Programming. In *Proc. of ILPS '93*, Vancouver, 1993. The MIT Press. 308

[7] F. Dignum and M. Greaves. Issues in agent communication. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 1–16. Springer, 2000. 307

[8] FIPA. Fipa 2000, fipa Query Interaction Protocol Specification. Technical report, FIPA (Foundation for Intelligent Physical Agents), November 2000. 306

[9] L. Giordano, A. Martelli, and C. Schwind. Specifying and Verifying Systems of Communicating Agents in a Temporal Action Logic. In *Proc. of the 8th Conf. of AI*IA*, LNAI. Springer, 2003. 310

[10] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *J. of Logic Programming*, 31:59–83, 1997. 302

[11] A. Mamdani and J. Pitt. Communication protocols in multi-agent systems: A development method and reference architecture. In *Issues in Agent Communication*, volume 1916 of *LNCS*, pages 160–177. Springer, 2000. 306, 310

[12] James H. Odell, H. Van Dyke Parunak, and Bernhard Bauer. Representing agent interaction protocols in UML. In *Agent-Oriented Software Engineering*, pages 121–140. Springer, 2001. http://www.fipa.org/docs/input/f-in-00077/. 309

[13] V. Patti. *Programming Rational Agents: a Modal Approach in a Logic Programming Setting*. PhD thesis, Dipartimento di Informatica, Università degli Studi di Torino, Italy, 2002. Available at `http://www.di.unito.it/~patti/`. 302, 304, 308

[14] F. Sadri, F. Toni, and P. Torroni. Dialogues for Negotiation: Agent Varieties and Dialogue Sequences. In *Proc. of ATAL'01*, Seattle, WA, 2001. 310

[15] S. Shapiro, Y. Lespéance, and H. J. Levesque. Specifying communicative multi-agent systems. In *Agents and Multi-Agent Systems - Formalisms, Methodologies, and Applications*, volume 1441 of *LNAI*, pages 1–14. Springer-Verlag, 1998. 310

# Distributed Intelligent Management of Active Networks

G. Di Fatta, S. Gaglio, G. Lo Presti, G. Lo Re, and I. Selvaggio

ICAR, Istituto di Calcolo e Reti ad Alte Prestazioni, C.N.R.
Universita' degli Studi di Palermo
viale delle Scienze, 90128 Palermo, Italy
{difatta,gaglio,lopresti,lore,selvaggio}@pa.icar.cnr.it

**Abstract.** This paper focuses on improving computer network management by the adoption of artificial intelligence techniques. A logical inference system has being devised to enable automated isolation, diagnosis, and even repair of network problems, thus enhancing the reliability, performance, and security of networks. We propose a distributed multi-agent architecture for network management, where a logical reasoner acts as an external managing entity capable of directing, coordinating, and stimulating actions in an active management architecture. The active networks technology represents the lower level layer which makes possible the deployment of code which implement teleo-reactive agents, distributed across the whole network. We adopt the Situation Calculus to define a network model and the Reactive Golog language to implement the logical reasoner. An active network management architecture is used by the reasoner to inject and execute operational tasks in the network. The integrated system collects the advantages coming from logical reasoning and network programmability, and provides a powerful system capable of performing high-level management tasks in order to deal with network fault situations.

**Keywords:** Distributed Intelligent Systems, Automated Reasoning, Situation Calculus, Network Management.

## 1 Introduction

The enormous growth of computer networks and the emerging technologies for network programmability make the research on network management an interesting challenge. Network management is a complex activity, which very often requires the human intervention to create action management plans, to coordinate network assets, and to face fault situations. Network management applications are traditionally centralized around a managing entity, like in the Simple Network Management Protocol (SNMP) [15] approach, and very often the managing entity is a simple interface to a human operator. Several distributed architectures for network management have been proposed to relieve the load from the central management station and to distribute control tasks by means

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 312–323, 2003.
© Springer-Verlag Berlin Heidelberg 2003

of the Active Networks technology [1], [2], [5], [17], [16]. Active networks introduce network dynamic programming and allow an easy deployment of "ad hoc" solutions in the active nodes on behalf of contingent management tasks. Given simple management tasks, such for instance multiple failure traps, data merging, automatic backup-link activation, the management architectures based on Active Networks make easy to deploy a distributed strategy. Nevertheless, for more complex tasks, it is still required the human intervention because only experts, who know the network complexity, can understand an high level management goal and plan a sequence of intermediate steps to reach the main objective. In a traditional network management environment, if a user asks to know the causes of a network failure, a network expert, according to some strategy, can query the network devices to trace the problem back to its original causes. Planning of actions, prediction and classification of events, diagnosis or explanation of failures, etc., are typical human activities that advanced Artificial Intelligence techniques are today able to provide. In our approach we adopt the Situation Calculus to represent a logical model of the network and the Golog programming language to implement a reasoner capable of accomplishing generic high-level management tasks. The logical reasoner allows not only the representation of particular network states, but also the representation of their evolution. The evolution, namely, can be represented as a transition process from a situation to another one and it is triggered by the execution of particular actions. In this paper we propose a two levels framework for network management, where the upper level is represented by a logical, centralized inference system acting as an external managing entity capable of directing, coordinating, and stimulating actions in an active management architecture. To this end, it exploits the capabilities of a distributed active management framework, which represents the lower level and makes possible the deployment of code across the whole network. This way, our management architecture exploits the potential of "doing" of the Active Networks technology conjugated with the potential of "planning", which is typical of the artificial intelligent systems [8], [3]. The originality of this work relies on two different aspects. Firstly, the architecture adopts a logic programming language to implement a high-level logical reasoner capable of producing failure diagnoses, or generating investigation plans to better define the decision scenario. Secondly, it is able to plan and carry out the information acquisition by exploiting the active network framework [2] that provides simple and effective tools to capture the right information in the appropriate places of the network. The remainder of this paper is structured in the following way. Section 2 introduces how the situation calculus can be adopted in our scenario to model the dynamism of network events. In section 3 we describe a Network Management Framework based on Active Networks whose functionalities perfectly answer the requirement of dynamic programmability imposed by our inferential engine. Section 4 illustrates the architecture of the distributed network fault management system and section 5 reports some experimental results. Finally, section 6 traces some conclusions.

## 2 Intelligent Network Management

Network Management [12], [14] involves several different functions grouped in five main areas: configuration, fault, performances, security, and accounting. Current management protocols approach these functional tasks in a rigid way, generally based on centralized monitoring, analysis, and control carried out by human operation staff. This paper proposes the adoption of a two levels framework, where a decision system behaves as managing entity, and active Network Management system executes the operational tasks. The upper level automation is made possible by the adoption of a logic-programming environment, which intrinsically owns special features that easily allow the achievement of tasks such as the decision of actions, the prediction and classification of events, the diagnosis or explanation of failures, etc. Intelligent network management requires a model of the network, which is able to capture both the cause-effect relationships and their dynamic nature (time varying relationships). This way, a logical inference process can use the system model to relate events that happen in the time-space to some other events which can be seen as their root causes. We adopt the situation calculus[9] to model the network and its dynamic evolution. The situation calculus is a logic language specifically designed for representing dynamically changing world. The management system we have designed can be classified as an expert system that adopts a case-based strategy [4]. It is an expert system since it owns a complete knowledge of the working environment. Namely, among the logic predicates it is necessary to provide ontological descriptions for all the entities which populate the external world, and for all their relationships. Moreover, we provided the system with the further capability of retrieving new knowledge on the basis of the current situation where the world lies: in particular, if the information available in the knowledge base is not sufficient to reach some deductive goals, new data are required and successively acquired by means of specific sensors positioned in opportune nodes of the network. The twofold nature of the system allowed us to achieve a simple and accurate monitoring of the managed network. In particular, we exploit the noticeable capabilities offered by the Reactive Golog [13] language as the specific reasoning environment adopted to implement the system. The adoption of Reactive Golog language is due to its noticeable expressiveness and to its capability of providing simple and linear frameworks to the programmers.

### 2.1 Reactive Golog

Reactive Golog is a logic programming language, which merges the flow control and procedural constructs of an Algol like language with the logic constructs of Prolog. A Reactive Golog interpreter is easy to implement under a Prolog interpreter and makes available all the capabilities of the situation calculus. The most important components of a Reactive Golog program are:

- Primitive Actions;
- Fluents;
- Primitive Actions Preconditions;
- Successor State Axioms;
- Procedures;
- Rules.

The formalization of the world is performed through well formed formulas of the first order logic, while the dynamism is captured through the primitive concepts of state, primitive action and fluent. We can think the state as a snapshot of the world at a determined moment. All changes to the world can be seen as the result of some primitive actions. Relations whose truth-values may vary in different situations are called relational fluents. They are represented by means of predicate symbols which take a situation term as their last argument. Primitive actions preconditions are rules that describe when actions can be carried out given a state of the world. The preconditions are stated by fluents. The successor state axioms provide a complete description about the fluents evolution in response to primitive actions. They are needed for each predicate that may change its truth value over the time. Procedures represent the complex actions and constitute one of the most important features of the Reactive Golog. They allow to group long sequences of primitive actions and to implement recursive formulas. Like in the imperative languages, they uses formal parameters. Generally, dynamic systems are not totally isolated by the rest of the world, but they continually receive solicitations and they interact with the external world. The Reactive Golog rules allow these interactions describing how the world evolves when an external action is performed. This aspect is the so called "reactive behavior".

### 2.2 An Example of Network Modeling

One of the key challenges of our work is the construction of a logic model capable of fitting as more as possible networking concepts. To this end, we have formalized all the entities which constitute the network layer of the OSI reference model, and all their basic capabilities. The ontological engineering process about which network aspects should be represented and which form of representation could present the most suitable features, induced us to establish a relationship between the functional layers of the OSI networking model and some correspondent layers of dynamic knowledge representation. In this vision, the lower level view concerns the physical features of the network, while, for instance, routing devices and their connecting communication links represent knowledge at a higher level. Furthermore, this first amount of knowledge representation has been integrated with the capability of representing the functioning during the time. Logical sentences, whose validity is bounded to the time, have been introduced for representing the temporal status of a given node or a particular link. In order to represent the network as a dynamic system capable of flowing from a situation (current state) to another one (successor state), we imagined the

network as an active entity capable of carrying out actions which modify its own configuration. According to the Situation Calculus, we selected the atomic actions that the network is able to carry out, and we described the state transitions involved by the actions execution. We defined some basic actions for managing the changes in the status of the basic network elements such as nodes, interfaces, links, sensors, active local agents (ALAs), routing tables, etc. Moreover a set of special purpose actions were defined to allow the notification of network monitored events to the logical reasoning system. These actions are performed by the network, thought as an active entity.

As previously described, the Situation Calculus allows the distinction among different temporal situations in which a single predicate may or may not be verified. This differentiation is achieved by means of the current situation as last argument of the fluents. For instance, the *Net_Link* and *Routing_table* predicates, respectively used to represent networks links and routing tables of the network routers, have to be written in the fluent form because their validity depends from the current situation. Here, in order to give an idea of the logical design process we produce a simple example. In the example we show as our logical system manages the routing tables update process for keeping traces of the changes in the networks. To this end, two primitive actions, define_rip_table(N,T) and update_rip_table(N, Upd) are shown.

**Primitive Actions**

```
primitive_action(define_rip_table(N,T)).
primitive_action(update_rip_table(N,Upd)).
```

The first action is used to instantiate the fluent predicate "routing_table(N,T,s)" in the situation s resulting from its own execution. This way, the logical reasoner will achieve the knowledge that, in the situation s, T is the routing table of the node N. The second action will update the same fluent. Namely, if the fluent "routing_table(N,T,s)" is verified, the update_rip_table(N,Upd) predicate will update the old value of the node N routing tables with the Upd information. The above actions require the following preconditions for their correct usage in the situation s of the network:

**Primitive Actions Preconditions**

```
poss(define_rip_table(N,T),S):- not routing_table(N,_,S).
poss(update_rip_table(N,Upd),S) :- routing_table(N,_,S).
```

The preconditions are determined by the truth-value assumed by the fluent predicate in the situation s and their validity enables the action executions. In the above example, the action define_rip_table(N,T) is possible in the situation s if the fluent routing_table(N,_,S) assumes a False value in this situation. Likewise for the action update_rip_table(N,Upd), which instead will be possible when the routing_table(N,_,S) fluent assumes a True value. The State Successor Axioms

are used to define the effects of the actions over the world, and their definition is necessary for each fluent. For instance the routing_table(N,T) fluent will assume a True value if the define_rip_table(N,T) action has occurred or if a previous routing_table(N,Old) has been modified by the update_rip_table(N,Upd) action.

**State Successor Axioms**

```
routing_table(N,T,do(A,S)) :-
        (routing_table(N,T,S), not A=update_rip_table(N,_)) ;
        ;((A = define_rip_table(N,T)) ;
          ;(A = update_rip_table(N,Upd),routing_table(N, Old,S),
                                                    tab_merge(Upd, Old,T))).
```

## 3 Active Networks Management Framework

In this section we describe the management architecture for Active Networks which has been adopted to perform the operational tasks required by the managing entity. Active Networks introduce programmability in the network; new software components can be dynamically injected in the network nodes. In our active management framework we exploit network programmability in order to support distributed, cooperative, and adaptive management applications.

The general framework is shown in figure 1. The logical reasoner, which operates as a Managing Entity (ME), sends queries and receives replies in the Extensible Markup Language (XML) to and from an AN Access Point. The AN Access Point is an active node hosting a Gateway service which performs two basic tasks: the translation of XML requests to the specific language adopted by the Execution Environment (EE) and the injection in the network of the appropriate active packets to accomplish ME requests. The Gateway services provide the interface to a particular active network implementation (EE) and it is specific for the supported network programming language. In general, several nodes in an active network can be configured to provide Gateway services. As

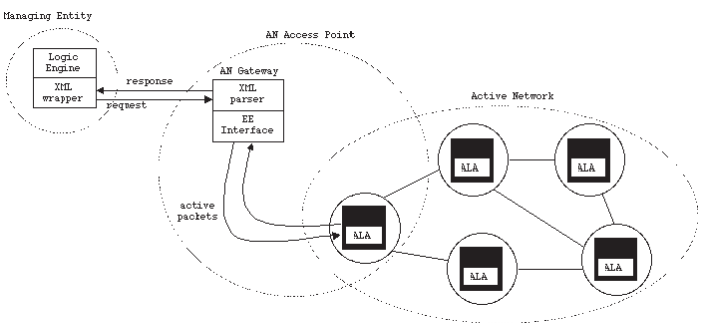



**Fig. 1.** *Active Network Management Framework*

illustrated in figure 1, a resident management service, the Active Local Agent (ALA), is resident in each active node. ME and ALA are the end points of the management communication. Namely, the ME can either query the Active Local Agents (polling) or deploy subtasks to them (programmable trapping). Local agents asynchronously perform subtasks in terms of actions to be executed at local events occurrences. The Active Local Agents can be seen as real teleo-reactive programs [11] which are installed and executed in the network nodes. Namely, the internal mechanism of ALAs involves the installation of predetermined local variables which implement the conditional statement of teleo-reactive agents. These variables constitute the discriminating values over which filters are installed in order to generate events, that in turn cause the execution of some actions. Finally, actions may install further conditions which will be evaluated till the predefined goal is achieved, i.e. the monitoring of fault root causes.

The architecture can be shared by different AN implementations and provides a common management framework for different EEs. We adopted XML to define a set of requests/replies for basic operational tasks, which are common to any Active Network environment. The Gateway basic services currently available are described in [2]. However, the set of Gateway services can be gradually enlarged. Once a new service is developed and tested, it can be provided for public use. Each basic service requires an EE-specific code fragment stored at the Gateway. This way, Gateway nodes provide transparent access to different Active Networks. For instance, the ME can use Gateway services to discover the network topology, explore the network nodes, find out which active applications are running and monitor their activities. We are particularly interested in managing those services, which are able to carry out specific tasks on behalf of the managing entity, such as the retrieval of a particular information from a node, or the verification of compound tests on several routers. The system dynamically deploys its components in the active nodes to implement a distributed strategy for a given goal. Two aspects are worth to be noticed: the reasoner adopts the active management architecture to exploit the programmability of network nodes and deploy ad-hoc tasks for a given goal; on the other hand the active management infrastructure can be used to manage, to monitor and to debug distributed network applications.

## 4 Network Faults Management System

The aim of our system, is related to the knowledge management of network events. To this end it deals with large archives of events obtained from the network systems, and extracts useful information from these data. Goals of such a system are:

- to diagnose root causes of network faults and performance degradations by establishing relationships between network events;
- to filter event (alarm) flood by correlating several events into a single conceptual event;
- to retrieve further ‘ad hoc’ diagnostic information;

– to adopt repair actions.

Such a system should provide:

– correctness: the root causes inferred by the system should be entailed by the detected events with a high likelihood, i.e. the root causes have really occurred in the network;
– optimality: the system should infer as small a set of root causes that can explain all the detected events.

The knowledge of network events constitutes the necessary information to answer questions about the real causes of network failures. Faults management is based on the definition of the normal operating conditions, and on the abnormalities detection. In general, alarms are generated in the network when abnormalities are detected. In alarm-based fault detection systems, a single fault will often cause a large number of alarms. Moreover, several faults may coexist causing a cascade of alarms. Our system is able to correlate alarms, to pinpoint their root causes in order to efficiently handle them. The logical inference process starts with a initial phase of information retrieval allowing the system to acquire the basic data over which can be established the successive reasoning processes. In analogy with an human behavior this step may correspond to the initial observation performed by an human operator to achieve an overall knowledge of the reasoning domain. The macroscopic behavior of the logical system is shown in the figure 2. In the following, the main steps involved in the monitoring of faults will be described. The process of faults management can be distinguished in a succession of steps [6] related to the alarm condition signaling, the error analysis with its consequential step of acquisition of additional data useful for a correct diagnosis, and finally the correct fault recognition and correction.

**Alarm Signaling** In this phase the signal of a network failure can be risen either by an user or by a sensor previously installed, which for instance may

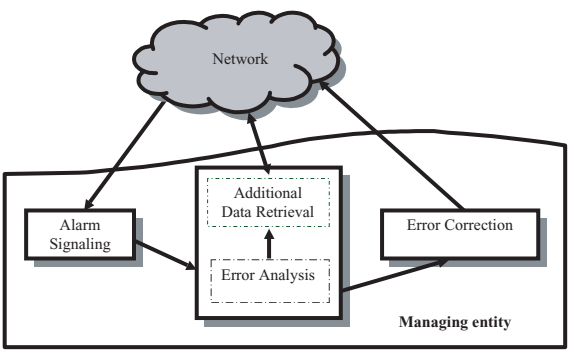



**Fig. 2.** *The logical inference process*

signal to the Managing Entity that a packet has been discarded on a router of the network.

**Error Analysis** Once the failure has been notified, the Managing Entity using the previously stored information tries an inferential process in order to determine the root cause of the failure. In the positive case, the fault is recognized and the successive step is performed. In the negative case, i.e. the knowledge base does not contain the right elements to deduce the root cause, the Managing Entity produces actions devoted to the positioning of new sensors with the aim to collect new diagnostic elements. The whole process is repeated until the Managing Entity is able to determine the failure nature.

**Error Correction** After the Error Analysis has produced the diagnostic inference, some predefined actions are taken in order to fix the discovered failure.
In the current version, the system is capable of managing the following types of events:

- Errors in the RIP routing tables of the network routers;
- Early packet discard for expired TTL;
- Full node failure;
- Full link failure;
- Loops in the RIP routing tables of the network routers;
- Changes of the state of neighbor routers.

To demonstrate the effectiveness of the automated reasoning system we analyze in more detail its behavior in a sample case of functioning. The capabilities of discovering the root causes of failure and of gathering additional data required by the reasoning process can be illustrated by the following example. Let us suppose that a routing loop has been generated on the network. The intervention of the Managing Entity can be triggered by a symptom, in this case packets discarding for expired TTL. The $primitive_action(lost_pkt_up(N))$ will notify this occurrence. Reasoning on the imagine routing tables maintained in the knowledge base, the ME is able to discover the loop occurrence. Namely, when a packet is discarded, the sensor installed on the involved router notifies the abnormal condition to the logical reasoner sending to it source and destination of the packet. Using this information the reasoner can simulate a journey trough the path from the source to the destination, using the routing tables of the encountered nodes. Throughout the simulation it marks each node traversed along the path, and if a previously marked node is met, the reasoner infers that a loop exits in the network. Furthermore, if throughout the simulation some routing table is missing, the reasoner may require these data to the opportune nodes by means of appropriate exogenous Golog actions, thus exploiting the Situation Calculus paradigm. Furthermore, in order to detect the node which is responsible of the failure, the actions taken by Managing Entity will consist of the routing tables changes retrieval by means of the following preocedure:

$$proc(collect(List), ?([] = List)\#?([X|Tail] = List) : get_local_var(X) : collect(Tail)).$$

In the above procedure List contains all the nodes which belong to the loop. In this procedure, firstly, it is considered the case of empty list. Otherwise, List is assumed as $[X|Tail]$, the external predicate $get_local_var(X)$ is invoked in order to obtain further information about the node $X$ routing table history, and finally the recursive call $collect(Tail)$ is executed. The collected changes are stored in the fluent $repository(_,_)$ which has the following state successor axiom.

$$\begin{aligned} repository(L, Do(A, S)) : & -repository(L, S), notA = update_repository(_,_,_); \\ & ; A = update_repository(N, [Dest, Neigh, Cost], Time), \\ & , repository(Old, S), L\ is[[N, [Dest, Neigh, Cost], Time]|Old]. \end{aligned}$$

This way, the system, by reasoning on the wider scenario enriched by the additional data, will be capable of singling out the root cause of the failure using the following two procedures:

$$\begin{aligned} & proc(select(List, Rep), ?([] = Rep)\#?([[N, [Dest, Neigh, Cost], Time]|Tail] = Rep) : \\ & \quad : (?(member(Neigh, List)) : add_causes(N, Time) : \\ & \quad : select(List, Tail)\#select(List, Tail))). \end{aligned}$$

and

$$\begin{aligned} & proc(check_cause(List), ?(repository(R)) : select(List, R) : ?(causes(C)) : \\ & \quad : select_old(C, [N, T])). \end{aligned}$$

The former is responsible of extracting from the repository all the changes that may have provoked the loop. The latter is capable of selecting the root initial cause which has produced the routing tables anomaly.
Finally, the $proc(set_table(N, [Dest, Neigh, Cost]))$ will be the action adopted by the Managing Entity for the correction of the routing entry.

## 5 Experimental Results

In order to test and evaluate our logical inference system we used an experimental cluster of 40 active network nodes which allows the setup of "ad hoc" topologies. In particular, we set up an experimental test-bed constituted by an active network with 20 nodes (routers and end-hosts) managed by the "pland" [7] execution environment and super-visioned by the reasoner by means of the AN-Gate [2] software package. We performed a series of experimental tests devoted to determine the reliability degree of our system, in terms of discovered faults and performed repair actions. Fault events are generated according to a Poisson distribution for their temporal occurrences, whereas we use a uniform distribution for their spatial positioning. The failures generated are those described in the previous section and for each of them a series of 25 trials has been executed.

Table 1 shows the results of the experiments. For each managed kind of failure we report the percentage of cases discovered and the average time elapsed before the failure recognition. Most of the events are captured the full percentage of the cases. In the cases of events directly discovered by the Active Local Agents on

**Table 1.** Summary of failure detection

| Failure | Discovered % | Avg. discov. time |
|---|---|---|
| Errors in the RIP routing tables | 96 | 10 sec. |
| Early Packet Discarding | 100 | 5 sec. |
| Full Link Failure | 100 | 30 sec. |
| Full Node Failure | 88 | 30 sec. |
| Loops in the RIP routing tables | 96 | 10 sec. |
| Changes in the state of neighbor routers | 100 | 20 sec. |
| Backup Link Recovery | 100 | 35 sec. |

the nodes, for instance the case of changes in the states of neighbor routers, the discovering times are constant since they depend on the sampling time of the monitoring activity. The low percentage for the case of full node failure is to be attributed to the simultaneous failures of several nodes. Namely, the occurrence of such events may provoke a disconnection in the network, thus denying the capability of retrieving the necessary alarms. Last row shows the measures of the action adopted to recover a link failure. The average time of 35 *sec.* is the overall time since the full link failure, thus meaning that the recovery time is reduced to 5 *sec.*. Finally, the overall percentage of false positives, also known as false detections or false alarms, has been measured and it is limited to the 1% of all the experimental cases.

## 6 Conclusions

The novelty of our project arises from the original idea of complementing a logical reasoner with the versatility of active networks. The integrated system collects the advantages coming from logical reasoning and network programmability, and realizes a powerful system capable of performing high-level management tasks and dealing with unusual network situations better than traditional management systems. The logical reasoner is, namely, able to deduce knowledge and find correlations from data and events which are distributed on different places of the networks and which occurred in different instants. The inferential engine provides reasoning on a high-level network model and behaves as an intelligent management agent, which coordinates the management activities at the low-level active network infrastructure. The framework allows the programming of intelligent management entities, which adopt the active networks management framework for the sensorial and actuator tasks, and the inferential logical system for the high-level behavior reasoner.

## References


[1] Al Shaer E., "Active Management Framework for Distributed Multimedia Systems", Journal of Networks and Systems Management, vol. 8 n. 1, 2000, pp.49-72. 313

[2] A. Barone, P. Chirco, G. Di Fatta, G. Lo Re, "A Management Architecture For Active Networks", Proc. of IEEE Workshop on Active Middleware Services. Edinburgh, UK, July 2002. 313, 318, 321

[3] Covo, A. A, Moruzzi, T. M., Peterson, E. D., "AI-assisted Telecommunications Network Management", In IEEE Global Telecommunications Conference (GLOBECOM 89), pages 487–496, Dallas, TX, USA, Nov 1989. 313

[4] Cebulka, K. D., Muller, M. J., and Riley, C. A. (1989). Applications of artificial intelligence for meeting network management challenges of the 1990s. In IEEE Global Telecommunications Conference (GLOBECOM 89), pages 501–506, Dallas, TX, USA, Nov 1989. 314

[5] Kawamura R., Stadler R., "Active Distributed Management for IP Networks", IEEE Communications Magazine, vol. 38, N. 4. April 2000, pp. 114-121. 313

[6] L. Kerschberg, R. Baum, A. Waisanen, I. Huang and J. Yoon, "Managing Faults in Telecommunications Networks: A Taxonomy to Knowledge-Based Approaches", IEEE, pp. 779-784, 1991 319

[7] Hick, M., et al, "PLAN: A Packet Language for Active Networks", Proc. of 3rd ACM SIGPLAN International Conference on Functional Programming, pages 86-93. ACM, September 1998. 321

[8] Mazumdar, S., Lazar, A., "Objective-Driven Monitoring for Broadband Networks", IEEE Transactions on Knowledge and Data Engineering, vol. 8, n. 3 june 1996, pp. 391 - 402 313

[9] J. McCarthy, "Situations, actions and causal laws", Technical Report Stanford University 1963. Reprinted in Semantic Information Processing, (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417 314

[10] Mills D. L., "On the accuracy of Clocks Synchronized by the Network Time Protocol in the Internet System", ACM Computer Communication Review, vol. 20, no. 1, pp. 65-75, Jan. 1990.

[11] Nilson N. J., "Intelligenza Artificiale", Apogeo, 2002 318

[12] Raman Lakshmi, "OSI Systems and Network Management", IEEE Communications Magazine, March 1998, Vol.36, N.3, pp.46-53. 314

[13] Reiter Raymond, "Knowledge in action: Logical Foundations for specifying and implementing Dynamical Systems" The MIT Press Cambridge Massachusetts 2001 314

[14] Sidor, D. J., "TMN standards: Satisfying Todays Need While Preparing for Tomorrow", IEEE Communications Magazine, March 1998, Vol.36, N.3, pp. 54-64. 314

[15] Stallings W., "SNMP and SNMPv2: The Infrastructure for Network Management", IEEE Communicatiions Magazine, vol. 36, N. 3 March 1998, pp. 37-45. 312

[16] Tennenhouse, D. L., Smith, J. M., Sincoskie, W. D., Wetherall D. J., Minde, G. J.: "A Survey of Active Network Research", IEEE Communications Magazine, Vol. 35, No. 1, January 1997, 8 313

[17] Tennenhouse, D. L., Wetherall, D. J.: "Towards an Active Network Architecture", Computer Communication Review, Vol. 26, No. 2, April 1996 313

# Planning under Uncertainty in Linear Time Logic*

Marta Cialdea Mayer, Carla Limongelli,
Andrea Orlandini, and Valentina Poggioni

Università di Roma Tre, Dipartimento di Informatica e Automazione
Via della Vasca Navale, 79, 00146 Rome, Italy
{cialdea,limongel,orlandin,poggioni}@dia.uniroma3.it

**Abstract.** The "planning as satisfiability" approach for classical planning establishes a correspondence between planning problems and logical theories, and, consequently, between plans and models. This work proposes a similar framework for contingency planning: considering contingent planning problems where the sources of indeterminism are incomplete knowledge about the initial state, non-inertial fluents and non-deterministic actions, it shows how to encode such problems into Linear Time Logic. Exploiting the semantics of the logic, and the notion of *conditioned model* introduced in this work, a formal characterization is given of the notion of contingent plan (a plan together with the set of conditions that ensure its executability).

## 1 Introduction

Planning under uncertainty and incomplete information is felt as an important issue in the planning community. In fact, classical planning, where the whole plan is synthesized off-line, is based on a number of simplifying assumptions: the world is completely known and predictable and no sensor information can be gathered at execution time. Such hypotheses however are too restrictive to model a realistic environment, where the description of the initial state may be incomplete, actions may have non-deterministic effects, and the agent interacts with the environment and gathers information from sensors.

In order to handle such sources of uncertainty, partial-order planning techniques have been extended to planning with incomplete information [15, 18]. Also logic-based approaches to planning have been investigated: for instance, [13] proposes a model based on deduction; [20] reduces planning to truth evaluation of quantified boolean formula; [14] extends the "planning as satisfiability" approach in propositional logic, originally introduced in [12]; [16] uses a logic of knowledge; [3] presents an approach based on model checking.

When aiming at solving problems defined in more complex and realistic domains, the need for more expressive planning languages is even stronger than in classical planning. Such a problem has been addressed, for instance, in [10, 21, 7].

* This work has been partially supported by ASI – Agenzia Spaziale Italiana.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 324–335, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Considering that, in this respect, logical approaches to planning are more flexible and expressive than others, this work proposes a logical framework for planning with uncertainty in Linear Time Logic (LTL), following the "planning as satisfiability" approach. This work extends [9], where classical planning, modeled in LTL, is reduced to model search. The model of time underlying LTL is a countably infinite sequence of states (or *time points*), that can be identified with $\mathbb{N}$. A temporal interpretation describes how the world evolves in time, hence a set of formulae constrains the behaviour of the world and can be taken as the specification of a planning domain. The advantages of using LTL instead of classical propositional logic, like in [12, 14], come from its greater expressive power. In fact, there is no need to represent time points explicitly, neither to fix a bound to the plan length in advance, and iteratively deepen such a bound in order to obtain a complete search procedure. The expressive power of LTL allows one also to represent domain restrictions, intermediate tasks, domain and control knowledge (useful to guide the search), as well as temporally extended goals [1, 2, 17].

This paper defines how to encode a planning problem, represented in a general planning language allowing for different forms of non-determinism (incomplete knowledge about the initial state, non-deterministic operators and non-determinism in the environment) and extended goals, into a set of LTL formulae (Section 2), thus extending the analogous translation of classical planning problems defined in [6] and used by the system presented in [9].

In order to deal with uncertain planning domains, the "planning as satisfiability" paradigm has to be adjusted, since a model always conveys complete information on what is true and what is false. This issue is addressed in the present work considering *conditioned models*, i.e. suitably structured partial descriptions of temporal interpretations, from which *contingent plans* can be extracted. A contingent plan is a a pair $\langle$*plan, conditions*$\rangle$, i.e. a plan together with the set of conditions that ensure its feasibility. Such a correspondence provides a formal characterization of the notion of contingent plan, through the semantics of LTL. In Section 3, the notion of conditioned model – already introduced in [8] – is revisited and extended in such a way that any contingent plan extracted from a conditioned model of a set $S$, representing a planning problem $P$, is a solution for $P$ (the corresponding version in [8] was in fact not adequate to the aims of planning, since it misses such a property). Finally, the notion of contingent plan is extended so that observations gathered during execution are taken into account. Such a characterization can be used when re-planning is needed.

## 2 A Planning Language for Non-deterministic Domains and Its Encoding in LTL

In this section, we define the kind of planning problems we are dealing with and their encoding in LTL. The problems are presented by means of a PDDL-like syntax. The language we use can be seen as an extension of the planning language used by the system presented in [9], that allows also for the specification of different forms of domain-specific knowledge, such as heuristic knowledge and

information on domain invariants. Such features are not considered in this work, since we focus here on the encoding of general planning problems. The language allows also for universal and existential quantification (over finite and fixed domains) – that are however treated as conjunctions or disjunction of propositional formulae –, but for simplicity we restrict here the language to be propositional.

The language we consider is built over an alphabet $\mathbf{P} = \mathcal{A} \cup \mathcal{F}$ of propositional letters, partitioned into the two sets $\mathcal{A}$ of action names and $\mathcal{F}$ of fluents. In turn, $\mathcal{F}$ is partitioned into the two sets $\mathcal{I}$ of inertial fluents and $\mathcal{F} - \mathcal{I}$ of non-inertial ones. Inertial fluents [11] are subject to the "common sense law of inertia": if no action affects them, their value stays unchanged. The value of a non-inertial fluent, on the contrary, cannot be always predicted from its previous value and the agent's actions, but it may be subject to exogenous events. It can also be affected by the agent's actions, but the effect of actions on non-inertial fluents is guaranteed for the next state only.

We consider the language of LTL over $\mathbf{P}$ built by means of the connectives $\neg$, $\wedge$, $\vee$ and the future time operators $\Box$ (always), $\Diamond$ (eventually), $\bigcirc$ (next). Implication, double implication and exclusive or ($\oplus$) are defined symbols.

To our aims, a non standard way of describing the semantics of LTL is more convenient [8]. It makes use of *labelled formulae*, i.e. expressions of the form $n : A$ where $n \in \mathbb{N}$ and $A$ is a formula. Intuitively, the label identifies the state where the formula is true. If $\ell$ is a literal, then $n : \ell$ is a labelled literal. A set $S$ of labelled literals is complete if for all $n \in \mathbb{N}$ and all $p \in \mathbf{P}$, either $n : p \in S$ or $n : \neg p \in S$. It is consistent if it does not contain both $n : p$ and $n : \neg p$ for any $n \in \mathbb{N}$ and $p \in \mathbf{P}$.

**Definition 1.** *A temporal interpretation $\mathcal{M}$ is a consistent and complete set of labelled literals.*

Intuitively, if $n : p \in \mathcal{M}$ then $p$ is true at state $n$, and if $n : \neg p \in \mathcal{M}$ then $p$ is false at state $n$.

If $\mathcal{M}$ is an interpretation of $\mathbf{P}$, $n \in \mathbb{N}$ and $A$ is a formula, the relation $\mathcal{M} \models n : A$ corresponds to the usual satisfiability relation $\mathcal{M}_n \models A$ (the $n$-th state of $\mathcal{M}$ satisfies $A$) and is recursively defined as follows:

$\mathcal{M} \models n : p$ iff $n : p \in \mathcal{M}$, when $p \in \mathbf{P}$;
$\mathcal{M} \models n : \neg A$ iff $\mathcal{M} \not\models n : A$;
$\mathcal{M} \models n : A \wedge B$ iff $\mathcal{M} \models n : A$ and $\mathcal{M} \models n : B$;
$\mathcal{M} \models n : A \vee B$ iff either $\mathcal{M} \models n : A$ or $\mathcal{M} \models n : B$;
$\mathcal{M} \models n : \bigcirc A$ iff $\mathcal{M} \models n + 1 : A$;
$\mathcal{M} \models n : \Box A$ iff for all $k \geq n$, $\mathcal{M} \models k : A$;
$\mathcal{M} \models n : \Diamond A$ iff for some $k \geq n$, $\mathcal{M} \models k : A$.

A formula $A$ is true in $\mathcal{M}$ (and $\mathcal{M}$ is a model of $A$) iff $\mathcal{M} \models 0 : A$. If $S$ is a set of formulae, $\mathcal{M} \models S$ iff $\mathcal{M} \models A$ for all $A \in S$. From now on, a set $S$ of LTL-formulae is considered the same as $\{0 : A \mid A \in S\}$.

The specification of a planning problem is divided into different sections. The declaration of fluents, partitioned into inertial and non-inertial, is contained in a specific section `fluents`.

The `init` section contains a set of classical formulae describing the initial state. Such a section may contain disjunctive information and is not required to be complete: all the unspecified information are supposed to be unknown, since the agent's knowledge of the initial state may be incomplete.

The goal of the problem is described in a corresponding section `goal`. The formula in this section may be a generic temporal formula, similarly to [17] (based on CTL), so that reachability goals as well as temporally extended goals can be encoded. For example, if $in$ is a fluent, then $\Diamond in$ is a reachability goal. The example 1 below provides an instance of a temporally extended goal: $\Diamond(in \wedge \Diamond\neg in)$, i.e. the agent has to enter the room and exit again. Goals can also have the form $\bigcirc\bigcirc\bigcirc in$ or $\Box\neg in$.

Each action is described in an `action` section, containing the operator's name, its preconditions and effects. Preconditions are classical formulae of any form. Conditional effects (`when <condition> <effect>`) are allowed, as well as non-deterministic ones, expressed as disjunctions or exclusive disjunctions of literals (atoms or negated atoms).

The following example shows how to specify non-inertial fluents and incomplete knowledge in the initial state:

*Example 1.* The agent has to enter a room, passing through a door, and exit again, without knowing whether the door is initially open or not. He can push the door open (from both sides), but the status of the door is not under the agent's complete control, for instance because other agents may pass by and open or close the door.

```
(:fluents (:non-inertial open)
          (:inertial in))
(:action pass
     :preconditions open
     :effects (and (when ¬in in)
                   (when in ¬in)))
```

```
(:action push
     :preconditions ¬open
     :effects open)
(:init ¬in)
(:goal ◇(in ∧ ◇¬in))
```

Here the execution of *pass* causes one to be inside the room if coming from outside, and to be out otherwise.

An example of non-deterministic action is the following:

*Example 2.* The agent has to get a file and print it. He does not know whether a postscript or pdf version of the document is available. Depending on this, he will use either Acrobat Reader or GhostView to print it.

```
(:fluents (:inertial (printed,have_ps,have_pdf)))
(:action gv
     :preconditions have_ps
     :effects printed)
(:action acroread
     :preconditions have_pdf
     :effects printed)
```

```
(:action take_file
     :preconditions ¬have_ps ∧ ¬have_pdf
     :effects have_ps ⊕ have_pdf)
(:init ¬have_ps, ¬have_pdf, ¬printed)
(:goal ◇printed)
```

Here *take_file* non-deterministically causes one to obtain a pdf or postscript file (and not both). The symbol $\oplus$ is used for exclusive OR.

In the rest of this section we show how to encode a planning problem into a set of LTL formulae. The encoding is an extension of the encoding of classical planning problems proposed in [6] and used in [9]. The specification of a planning problem in the language described above is encoded by the set $S_0$ of LTL formulae obtained as follows.

**Initial State Axioms:** All the formulae describing the initial state (in the `init` section) are included in $S_0$. This results in a (possibly incomplete) description of the initial state.

**Goal Axioms:** The encoding of the goal $G$ (stated in the `goal` section) is $G$ itself.

**Action Precondition Axioms:** For each operator $a$ having preconditions $A_1, ..., A_n$, $S_0$ contains the formula: $\Box(a \rightarrow A_1 \wedge ... \wedge A_n)$.

**Non-deterministic Effect Axioms:** For each action $a$, having disjunctive effects $\ell_1 \star ... \star \ell_k$, where $\star$ is either $\vee$ or $\oplus$, under conditions $C$, (`when` $C$ $(\ell_1 \star ... \star \ell_k)$) $S_0$ contains: $\Box(a \wedge C \rightarrow \bigcirc\ell_1 \star ... \star \bigcirc\ell_k)$ (if the effect is unconditioned, $C$ is omitted).

**Effect Axioms:** If $f$ is a non-inertial fluent or a fluent that can be *non deterministically* affected by some action, let $G_f^+$ be the disjunction of all the $a_i \wedge C_i$ that have $f$ as a deterministic effect ($G_f^+$ specifies all the conditions that *deterministically* lead to change the truth value of $f$ from false to true), and $G_f^-$ the disjunction of all the $a_i \wedge C_i$ that have $\neg f$ as a deterministic effect ($G_f^-$ specifies all the conditions that *deterministically* lead to change the truth value of $f$ from true to false). Then $S_0$ contains the following effect axioms for each of such fluents:

$$\Box(G_f^+ \rightarrow \bigcirc f), \Box(G_f^- \rightarrow \bigcirc\neg f) \tag{1}$$

**Inertia Axioms:** For each inertial fluent $f$, such that:

(i) $\neg f$ is a non-deterministic effect of some actions $a_1, .., a_k$, under conditions, respectively, $C_1, ..., C_k$;

(ii) $f$ is a non-deterministic effect of some actions $b_1, .., b_n$, under conditions, respectively, $D_1, ..., D_n$,

$S_0$ contains the following axioms:

$$\Box((f \wedge \neg G_f^- \wedge \neg(a_1 \wedge C_1) \wedge ...\neg(a_k \wedge C_k)) \rightarrow \bigcirc f) \tag{2}$$
$$\Box(\bigcirc f \rightarrow (f \vee G_f^+ \vee (b_1 \wedge D_1) \vee ... \vee (b_n \wedge D_n)))$$

where $G_f^+$ and $G_f^-$ are as specified in the previous item and they refer to the action that have $f$ or $\neg f$ as a deterministic effect.

**Successor State Axioms:** For each inertial fluent $f$, that is never a non-deterministic effect of any action, $S_0$ contains the following paraphrases of Reiter's Successor State Axiom [19]:

$$\Box(\bigcirc f \equiv G_f^+ \vee (f \wedge \neg G_f^-))$$

**Incompatibility Axioms:** $S_0$ contains also a set of axioms describing incompatibility relations between actions, of the form $\Box(\neg a \vee \neg b)$. What has to

be explicitly encoded, however, are the incompatibilities of actions $a$ and $b$ such that $a$ deletes a precondition of $b$, or vice-versa. In fact, if two actions $a$ and $b$ have conflicting preconditions or effects, such an incompatibility is taken into account by the logic itself, since $a$ and $b$ cannot be true in the same state. For convenience, we also declare two actions $a$ and $b$ to be incompatible whenever $a$ has a non-deterministic effect $f$ and $b$ a (deterministic or non-deterministic) effect $\neg f$ (or vice-versa).

The formulae (1) e (2) correspond to the Successor State Axioms for non-inertial fluents or fluents non-deterministically affected by some action.

The adequacy of the analogous encoding for classical planning problems has been shown in [6]. In the new setting, the encoding can be recognized correct when we consider that a successor state axiom is equivalent to the conjunction of the following formulae:

$$(i)\ \ \Box(G_f^+ \rightarrow \bigcirc f) \qquad (iii)\ \Box(f \wedge \neg G_f^- \rightarrow \bigcirc f)$$
$$(ii)\ \Box(G_f^- \rightarrow \bigcirc \neg f) \qquad (iv)\ \Box(\neg f \wedge \neg G_f^+ \rightarrow \bigcirc \neg f)$$

Non-inertial fluents clearly do not satisfy $(iii)$ and $(iv)$, while $(i)$ and $(ii)$ are present in $S_0$ as effect axioms. If $f$ is a non-deterministic effect of some action, $(iii)$ and $(iv)$ are in $S_0$ as inertia axioms, $(i)$ and $(ii)$ are replaced by the effect and non-deterministic effect axioms.

Note that, differently from [5], an action name is true in the state where the action is performed, and not in the resulting state. In this way, there is no need to characterize at the meta-level the notion of "legal plan".

Figure 1 shows the complete encoding of the two examples presented before, filtered by the application of some simple logical simplifications.

| *LTL encoding of Example 1* | *LTL encoding of Example 2* |
|---|---|
| **Initial state axioms:**<br>$\neg in$<br>**Goal axioms:**<br>$\Diamond(in \wedge \Diamond\neg in)$<br>**Action preconditions axioms:**<br>$\Box(pass \rightarrow open)$,<br>$\Box(push \rightarrow \neg open)$<br><br>**Fluents behaviour:**<br>$\Box(push \rightarrow \bigcirc open)$,<br>$\Box(\bigcirc in \equiv$<br>$(in \wedge \neg pass) \vee (\neg in \wedge pass))$ | **Initial state axioms:**<br>$\neg have_ps,\ \neg have_pdf,\ \neg printed$<br>**Goal axioms:**<br>$\Diamond printed$<br>**Action preconditions axioms:**<br>$\Box(take_file \rightarrow \neg have_ps \wedge \neg have_pdf)$<br>$\Box(acroread \rightarrow have_pdf)$<br>$\Box(gv \rightarrow have_ps)$<br>**Fluents behaviour:**<br>$\Box(take_file \rightarrow (\bigcirc have_ps \wedge \bigcirc\neg have_pdf)$<br>$\vee(\bigcirc have_pdf \wedge \bigcirc\neg have_ps)))$<br>$\Box(have_ps \rightarrow \bigcirc have_ps)$<br>$\Box(have_pdf \rightarrow \bigcirc have_pdf)$<br>$\Box(\bigcirc have_ps \rightarrow have_ps \vee take_file)$<br>$\Box(\bigcirc have_pdf \rightarrow have_pdf \vee take_file)$<br>$\Box(\bigcirc printed \equiv (gv \vee acroread \vee printed))$ |

**Fig. 1.** The complete encodings of examples 1 and 2

## 3 Contingent Plans

In this section, we identify a planning problem with its temporal logic encoding and formalize the notion of contingent plan in terms of temporal interpretations.

Instead of producing the whole conditional plan off-line, and then executing it, we can consider the possibility that planning and execution (with sensing actions) are interleaved. In that case, a single contingent plan can be generated at first. During its execution, its conditions are checked by means of sensing actions. When a condition turns out to be false, a different contingent plan is generated and considered. Obviously, such a procedure is complete only if all actions are reversible. Such a restricion is discussed in Section 4.

Let $S_0$ be a set of formulae encoding a planning problem, as illustrated before. From a model $\mathcal{M}$ of $S_0$ we want to extract a contingent plan solving the problem represented by $S_0$, i.e. a significant subset of $\mathcal{M}$, describing a plan, together with a set of conditions ensuring the feasibility of the plan. In general, in fact, it is not the whole model $\mathcal{M}$ that is of interest and has to be explained, but only a finite fragment of the model, up to the achievement of the goals. Even the description of $\mathcal{M}$ up to such a "final" state need not be complete. However, it must be rich enough: it must contain enough information on how to reach the goal and which conditions ensure its reachability. The following definition formalizes this concept. For simplicity we assume here that $S_0$ contains formulae in negation normal form. If $S$ is a set of labelled formulae, then $lits(S)$ is the set of labelled literals in $S$.

**Definition 2.** *Let $S_0$ be a set of labelled formulae, $S \supseteq S_0$ and $m \in \mathbb{N}$. Then $S$ is a* saturated extension *of $S_0$ up to $m$ if for every $n : A \in S$ such that $n \leq m$:*

1. *if $A = A_0 \wedge A_1$, then both $n : A_0 \in S$ and $n : A_1 \in S$;*
2. *if $A = A_0 \vee A_1$, then either $n : A_0 \in S$ or $n : A_1 \in S$;*
3. *if $A = \Box A_0$ then for all $k$ such that $n \leq k \leq m$, $k : A_0 \in S$;*
4. *if $A = \Diamond A_0$ then for some $k$ such that $n \leq k \leq m$, $k : A_0 \in S$;*
5. *if $A = \bigcirc A_0$ and $n < m$ then $n + 1 : A_0 \in S$.*

*If $\mathcal{M}$ is a model of $S_0$ and $S$ a saturated extension of $S_0$ up to $m$, such that $lits(S) \subset \mathcal{M}$, then $lits(S)$ is called a* significant description of $\mathcal{M}$ as a model of $S_0$.

Note that, mainly because of clause 4 above, any significant description of a model of the encoding of a planning problem implies that the goals are achieved.

It can easily be proved, by use of the same techniques used to prove soundness of tableau methods, that if $\mathcal{M}$ is a model of $S_0$ then there exists a significant description of $\mathcal{M}$ as a model of $S_0$.

The following definitions are introduced to the aim of characterizing an *adequate* set of conditions ensuring the feasibility of a plan.

**Definition 3.** *If $S$ is a set of labelled literals the* history *of $S$ up to $n$, $h(S, n)$, is*

$$\{k : \ell \mid k : \ell \in S \text{ and } k \leq n\}$$

*Let $S_0$ be a set of labelled formulae, and $K$ and $U$ sets of labelled literals. Then $U$ is a* set of conditions *explaining $K$ in the context of $S_0$ (briefly, $U$ explains $K$ in $S_0$) iff $U$ is a minimal set (w.r.t. set inclusion) such that:*

$$\textit{for all } n : \ell \in K: \;\; S_0, h(U, n-1) \models n : \ell.$$

The condition above amounts to saying that, for all $n : \ell \in K$ and for every model $\mathcal{M}'$ of $S_0$, if $\mathcal{M}'$ is also a model of the subset $h(U, n-1)$ of the history of $U$ up to $n-1$, then $\mathcal{M}' \models n : \ell$, too. In informal terms, $h(U, n-1)$ "explains", in the context of $S_0$, all that happens, according to $K$, up to the $n$-th state. Moreover, no subset of $U$ has such a property. If $K$ implies, in the context of $S_0$, that the goals are achieved within the $n$-th step, then $h(U, n-1)$ actually guarantees the achievement of the goals.

**Definition 4.** *Let $S_0$ be a set of labelled formulae representing a planning problem. A* conditioned model *of $S_0$ is a pair $\langle K, U\rangle$, where $K \cup U$ is a significant description of some model $\mathcal{M}$ of $S_0$ and $U$ explains $K$ in $S_0$.*

*If $\langle K, U\rangle$ is a conditioned model of $S_0$ then the contingent plan corresponding to $\langle K, U\rangle$ solving the problem represented by $S_0$ is the pair $\langle \mathcal{P}, \mathcal{C}\rangle$, where:*

$$\begin{aligned} \mathcal{P} &= \{n : a \mid n : a \in U \textit{ and } a \in \mathcal{A}\} \\ \mathcal{C} &= \{n : p \mid n : p \in U \textit{ and } p \in \mathcal{F}\} \cup \{n : \neg p \mid n : \neg p \in U \textit{ and } p \in \mathcal{F}\} \end{aligned}$$

In simpler words, a contingent plan solving the problem encoded by $S_0$ can be extracted from the set $U$ of conditions of a conditioned model of $S_0$, by dropping negative action literals, i.e. labelled literals of the form $n : \neg a$ for $a \in \mathcal{A}$, and splitting the rest of $U$ into the set of action names and the fluents (contingencies guaranteeing the feasibility of the plan).

For instance, let $S_1$ be the encoding of the problem presented in Example 1, and $S$ the following subset of one of its models:

$$\begin{aligned} S = \{ & 0 : \mathit{open},\ 0 : \mathit{pass},\ 0 : \neg\mathit{push},\ 0 : \neg\mathit{in}, \\ & 1 : \neg\mathit{open},\ 1 : \neg\mathit{pass},\ 1 : \mathit{push},\ 1 : \mathit{in}, \\ & 2 : \mathit{open},\ 2 : \mathit{pass},\ 2 : \neg\mathit{push},\ 2 : \mathit{in}, \\ & 3 : \mathit{open},\ 3 : \neg\mathit{pass},\ 3 : \neg\mathit{push},\ 3 : \neg\mathit{in}\} \end{aligned}$$

$S$ is a significant description of the model up to the third state; in particular it implies that the goals are achieved at the third state. Let us partition $S$ into:

$$\begin{aligned} U = \{ & 0 : \mathit{open},\ 0 : \mathit{pass},\ 0 : \neg\mathit{push}, 1 : \neg\mathit{open},\ 1 : \neg\mathit{pass},\ 1 : \mathit{push}, \\ & 2 : \mathit{pass},\ 2 : \neg\mathit{push},\ 3 : \mathit{open},\ 3 : \neg\mathit{pass},\ 3 : \neg\mathit{push}\} \\ K = \{ & 0 : \neg\mathit{in},\ 1 : \mathit{in},\ 2 : \mathit{open},\ 2 : \mathit{in},\ 3 : \neg\mathit{in}\} \end{aligned}$$

Then $U$ explains $K$ in $S_1$. In fact:

$$\begin{aligned} & h(U, -1) = \emptyset \\ & h(U, 0) = \{0 : \mathit{open},\ 0 : \mathit{pass},\ 0 : \neg\mathit{push}\} \\ & h(U, 1) = h(U, 0) \cup \{1 : \neg\mathit{open},\ 1 : \neg\mathit{pass},\ 1 : \mathit{push}\} \\ & h(U, 2) = h(U, 1) \cup \{2 : \mathit{pass},\ 2 : \neg\mathit{push}\} \end{aligned}$$

and

$$S_1 \models 0 : \neg in \qquad S_1,\ h(U,1)\ \models 2 : in$$
$$S_1,\ h(U,0)\ \models 1 : in \qquad S_1,\ h(U,2)\ \models 3 : \neg in$$
$$S_1,\ h(U,1)\ \models 2 : open$$

Omitting the negative "action" literals, $U$ can be read as saying:

IF $0 : open$, $0 : pass$, $1 : \neg open$, $1 : push$, and $2 : pass$,
THEN the goal is achieved.

The contingent plan that is extracted from $U$ is then the pair:

$$\langle\{0 : pass,\ 1 : push,\ 2 : pass\},\{0 : open,\ 1 : \neg open\}\rangle$$

Note that, according to Definition 4, the success of the plan could also be conditioned by contingencies regarding the same state where the goal is achieved. For instance, let us consider the following simplification of the problem presented as Example 2:

```
(:fluents
   (:inertial have_ps,have_pdf))
(:init ¬have_ps, ¬have_pdf)
(:goal ◇have_ps)
```

```
(:action take_file
   :preconditions
            ¬have_ps ∧ ¬have_pdf
   :effects have_ps ⊕ have_pdf)
```

A conditioned model of its encoding is:

$$K = \{0 : \neg have_ps,\ 0 : \neg have_pdf\}$$
$$U = \{0 : take_file,\ 1 : \neg have_pdf,\ 1 : have_ps\}$$

The goal *have_ps* is achieved at state 1, but it is conditioned by contingencies related to state 1 itself. In this case, although for $n = 1$, $S_0, U \models n : have_ps$, the achievement of the goal is not guaranteed only by $S_0 \cup h(U, n-1)$, but the whole set $U$ has to be taken into account.

Note that both $1 : have_ps$ and $1 : \neg have_pdf$ occur as conditions in $U$, although they are inter-derivable in the context of $S_0$; in fact, none of the two literals is derivable from $S_0$ and $h(U,0)$, since, at that step, the effect of $0 : take_file$ cannot be predicted. If this situation is to be excluded, then we must also require for a pair $\langle K, U\rangle$ to be a conditioned model of the encoding $S_0$ of a problem, that $S_0, K \models 0 : G$, where $G$ is the representation of the goal.

Let $\langle K, U\rangle$ be a conditioned model of the encoding $S_0$ of a planning problem and let us assume that, during the execution of the contingent plan extracted from $\langle K, U\rangle$, the set $\mathcal{O}$ of observations has been gathered. Let us assume, moreover, that $n : \ell$ is the result of the last sensing action and the only observation that is inconsistent with $U$. Then we want to synthesize a plan that is "complementary" to the previous one with respect to $n : \ell$. The new plan must take $\mathcal{O}$ into account, i.e. the observations, and whatever is derivable from them and $S_0$, should not appear as conditions in the new plan but they will be considered as known literals in that step.

**Definition 5.** *Let $S_0$ be the encoding of a planning problem, and $\mathcal{O}$ a set of observations, i.e. a set of labelled literals that is consistent with $S_0$. A conditioned model of $S_0$ in the presence of $\mathcal{O}$ is a conditioned model of $S_0 \cup \mathcal{O}$, i.e. a pair $\langle K, U\rangle$ such that $K \cup U$ is a significant description of $S_0 \cup \mathcal{O}$ and for all $n : \ell \in K$: $S_0, \mathcal{O}, h(U, n-1) \models n : \ell$.*

*Let $\langle K, U\rangle$ be a conditioned model of $S_0$, $\mathcal{O}$ a set of observations, and $n$ the maximum label of the literals in $\mathcal{O}$. Let us assume, moreover, that $n : \ell$ is the only labelled literal that is inconsistent with $U$. Then a pair $\langle K', U'\rangle$ is complementary to $\langle K, U\rangle$ in the presence of $\mathcal{O}$ if it is a conditioned model of $S_0$ in the presence of $\mathcal{O}$ and $K \cup U$ and $K' \cup U'$ describe the same models up to step $n-1$, i.e. for all $k < n$: $\{k : \ell \mid k : \ell \in K \cup U\} = \{k : \ell \mid k : \ell \in K' \cup U'\}$.*

For instance, let $S_2$ be the encoding of the problem presented as Example 2, where the *take_file* action has a non-deterministic effect, and let $\langle K, U\rangle$ be the conditioned model of $S_2$ where:

$$\begin{aligned} K = \{ & 0 : \neg have_ps,\ 0 : \neg have_pdf,\ 0 : \neg printed,\ 0 : \neg gv,\ 0 : \neg acroread, \\ & 1 : \neg printed,\ 1 : \neg take_file, \\ & 2 : printed,\ 2 : have_ps,\ 2 : \neg have_pdf\ ,\ 2 : \neg acroread,\ 2 : \neg take_file\} \\ U = \{ & 0 : take_file, 1 : have_ps, 1 : \neg have_pdf, 1 : gv, 1 : \neg acroread, 2 : \neg gv\} \end{aligned}$$

Let us assume that, during the execution of this plan, after performing the *take_file* action at step 0, *have_ps* turns out to be false at step 1, so the observation $\mathcal{O} = \{1 : \neg have_ps\}$ must be taken into account when synthesizing the new plan. A suitable new plan is unconditioned:

$$\langle\{0 : take_file,\ 1 : acroread\},\ \emptyset\rangle.$$

In fact, this is the plan corresponding to the conditioned model $\langle K', U'\rangle$ of $S_2 \cup \mathcal{O}$, where:

$$\begin{aligned} K' = \{ & 0 : \neg have_ps,\ 0 : \neg have_pdf,\ 0 : \neg printed, 0 : \neg gv,\ 0 : \neg acroread, \\ & 1 : \neg printed,\ 1 : \neg have_ps,\ 1 : have_pdf, 1 : \neg gv,\ 1 : \neg take_file, \\ & 2 : printed,\ 2 : \neg have_ps,\ 2 : have_pdf, 2 : \neg gv,\ 2 : \neg take_file\ \} \\ U' = \{ & 0 : take_file,\ 1 : acroread,\ 2 : \neg acroread\ \} \end{aligned}$$

In fact, $S_2, \mathcal{O}, 0 : take_file \models 1 : have_pdf$.

## 4 Concluding Remarks

The "planning as satisfiability" approach for classical planning establishes a correspondence between planning problems and logical theories, and, consequently, between plans and models. This work proposes a similar framework for contingency planning: considering contingent planning problems where the sources of indeterminism are incomplete knowledge about the initial state, non-inertial fluents and non-deterministic actions, it shows how to encode such problems into LTL. Exploiting the semantics of the logic, and the notion of conditioned model, we give a formal characterization of the notion of contingent plan.

A prototype system is under development, that simulates the interleaving of planning, execution and monitoring and it is based on the tableau calculus presented in [8]. We are presently testing the system and gathering experimental data.

As already remarked, the procedure based on a planning - executing - sensing - re-planning loop is complete (the goal is reached whenever possible) only under the assumption that actions are reversible. Such a limit, however, can be overcome in a context where control knowledge can be easily encoded. In fact, domain specific heuristic information can prevent dead-ends and guarantee the achievement of the goal, exploiting a good (domain-dependent) policy in the choice of actions.

Much work still remains to be done: from a practical point of view the prototype must be improved, in order to prove its effectiveness. The performances of the classical planning system presented in [9], that is based on the "planning as satisfiability (in LTL)" approach, have been shown to be comparable with other systems. This is mainly due to an efficient implementation of propositional subformulae, exploiting ROBDD [4]. The prototype, that at present uses a naive representation of formulae, has to be re-implemented in order to benefit from that technique. Once this is done, the system has to be completed with the other distinguishing features of this approach, i.e. the possibility to include heuristic knowledge and information about domain invariants, as well as the equipment with utility tools, such as consistency and redundancy checks, that can easily be done when the problem representation is entirely logical.

From the theoretical standpoint, an important issue to be addressed concerns the possibility to synthesize looping plans on the same basis. In fact, when linear plans are looked for, plan search stops as soon as a state $s_1$ is reached whose temporal description is identical to an already existing one, $s_0$. On the contrary, if loops are allowed, such situations have to be handled differently: the sequence of steps leading from $s_0$ to $s_1$ is recorded as a loop, and the search proceeds in a different direction. Some non-trivial work has however to be done in order to identify the correct conditions to exit the loop, and include them among the conditions of the contingent plan.

## References

[1] F. Bacchus and F. Kabanza. Using temporal logic to control search in a forward chaining planner. In *Proc. of the TIME-95 International Workshop on Temporal Representation and Reasoning*, Melbourne, Florida, April 1995. 325

[2] Fahiem Bacchus and Froduald Kabanza. Planning for temporally extended goals. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1215–1222. AAAI Press / The MIT Press, 1996. 325

[3] P. Bertoli, A. Cimatti, M. Roveri, and P. Traverso. Planning in non deterministic domains under partial observability via symbolic model checking. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence IJCAI 2001*, pages 473–478, 2001. 324

[4] Randal E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986. 334

[5] D. Calvanese, G. De Giacomo, and M. Y. Vardi. Reasoning about actions and planning in LTL action theories. In *Proc. of the 8th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'02)*, pages 593–602, 2002. 329

[6] S. Cerrito and M. Cialdea Mayer. Using linear temporal logic to model and solve planning problems. In F. Giunghiglia, editor, *Proceedings of the 8th International Conference on Artificial Intelligence: Methodology, Systems, Applications (AIMSA'98)*, pages 141–152. Springer, 1998. 325, 328, 329

[7] A. Cesta and A. Oddi. DDL.1: a formal description of a constraint representaton language for physical domains. In M. Ghallab and A. Milani, editors, *New Direction in AI Planning*, pages 341–352. IOS Press, 1996. 324

[8] M. Cialdea Mayer and C. Limongelli. Linear time logic, conditioned models and planning with incomplete knowledge. In C. Fermüller and U. Egly, editors, *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods*, volume 2381 of *LNAI*, pages 70–84. Springer, 2002. 325, 326, 334

[9] M. Cialdea Mayer, A. Orlandini, G. Balestreri, and C. Limongelli. A planner fully based on linear time logic. In S. Chien, S. Kambhampati, and C. A. Knoblock, editors, *Proc. of the 5th Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, pages 347–354. AAAI Press, 2000. 325, 328, 334

[10] K. Currie and A. Tate. O-Plan: the open planning architecture. *Journal of Artificial Intelligence*, 52:49–86, 1991. 324

[11] E. Giunchiglia, G. N. Kartha, and V. Lifschitz. Representing action: Indeterminacy and ramifications. *Artificial Intelligence*, 95(2):409–438, 1997. 326

[12] H. Kautz and B. Selman. Planning as satisfiability. In B. Neumann, editor, *10th European Conf. on Artificial Intelligence*, pages 360–363. Wiley & Sons, 1992. 324, 325

[13] H. Levesque. What is planning in the presence of sensing? In *Proc. of the 13th National Conference on Artificial Intelligence, AAAI-96*, pages 1139–1146. AAAI Press, 1996. 324

[14] E. Giunchiglia P. Ferraris. Planning as satisfiability in nondeterministic domains. In *Proc. of the Nat. Conf. on Artificial Intelligence (AAAI'00)*, 2000. 324, 325

[15] M. A. Peot and D. E. Smith. Conditional nonlinear planning. In J. Hendler, editor, *Proc. of the First International Conference on Artificial Intelligence Planning Systems*, pages 189–197. Morgan Kaufmann, 1992. 324

[16] R. P. A. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *Proc. of the 6th Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-2002)*, 2002. 324

[17] M. Pistore and P. Traverso. Planning as model checking for extended goals in non-deterministic domains. In *Proc. IJCAI'01*. AAAI Press, 2001. 325, 327

[18] L. Pryor and G. Collins. Planning for contingencies: a decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996. 324

[19] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and mathematical theory of computation: Papers in honor of John McCarthy*, pages 359–380. Academic Press, 1991. 328

[20] J. Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intellingence Research*, 10:323–352, 1999. 324

[21] B. Stephan and S. Biundo. Deduction based refinement planning. In B. Drabble, editor, *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 213–220. AAAI Press, 1996. 324

# Planning with Fuzzy Resources

Marco Baioletti[1], Alfredo Milani[2], and Valentina Poggioni[3]

[1] Dip. di Metodi Quantitativi, Università di Siena
P.zza San Francesco 8, 53100 Siena, Italy
asterix@stat.unipg.it
[2] Dip. Matematica e Informatica, Università di Perugia
Via Vanvitelli 1, 06123 Perugia, Italy
milani@dipmat.unipg.it
[3] Dip. di Informatica e Automazione, Università di Roma Tre
Via della Vasca Navale 79, 00146 Roma, Italy
poggioni@dia.uniroma3.it

**Abstract.** In this paper a model for planning with fuzzy resources is presented. Nearly all existing models of planning with resources need to work with exact value of the resources modified by actions execution. Therefore these models cannot deal with more realistic situations in which the resources quantities are not completely known. The model presented here allows to manage domains more tailored to real world, where preconditions and effects over resources can be specified in terms of uncertain intervals using fuzzy sets. Moreover it is possible to define problems with mixed logical/quantitative and pure numerical goals. The fuzzy approach adequately represents the kind of uncertainty existing in planning domains. The choice of trapezoidal fuzzy intervals represents a solution which combine a simple, but expressive representation, with computational simplicity. It is also presented a search strategy for goals over resources that provides a specific method to guide search. This strategy is necessary when solving "pure numerical problems". Finally some experimental results are presented and commented.

## 1 Introduction

Various models have been proposed for extending the pure logical classical planning models in order to manage more realistic world features. A very promising research line is represented by Planning with Resources. In this framework, in addition to the logical relationships among domain objects, operators and states, the planning model can handle quantitative aspects of the world, such as actions which involves consumable/reusable resources, domain constraints on resources and goals involving quantities.

Several planning models for resources management have been proposed for extending virtually all the most successful planner approaches; among them it is worth noticing UCPOP–like [2], Graphplan–like [8], SAT–like [13, 12], and also HTN based approaches [3].

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 336–348, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The types and the features of the modelled resources are also varying from unary and discrete resources [9] to reusable resources [2] and conjunctive constraints over resources [6, 4].

A simple extension is to allow for resources preconditions and effects, while keeping only logical goals. In a deeper extension, it is possible to specify "mixed" logical/quantitative goals or even "pure quantitative goals" (e.g. consider the problem of finding a plan for the purely quantitative goals: Consume at least 100 calories, Produce $1 Billion profit etc.). Finally pure/mixed "optimization" goals can also be specified, where no logical or quantitative goals exist (e.g. consider the problem: "producing as much profit as you can"). This latter optimization aspect has been incorporated in PDDL 2.1 [5], where it is possible to specify an object function to be optimized.

Models of planning with resources certainly represent an important step toward a more accurate model of the real world, but most of the proposed models fail to give any account of the potential imprecision which can affect the quantities related to resources. Many facts in a real world state can be described in a satisfactory way by a boolean proposition, but it is not very realistic to assume that an exact number can model the continuous quantities describing a given resource.

It is worth noting that the indetermination on resources cited in the preconditions or goals, such as numerical intervals or fuzzy sets, is not problematic: this situation is similar to the one with disjunctive preconditions and goals.

On the other hand, the indetermination on resources cited in the effects or initial state causes the planner to work in an underspecified world with actions that have a limited sort of non deterministic effects.

Thus a model with uncertain or fuzzy effects and initial state is a non deterministic one. Even the concept of a plan solution should be adapted: A plan is a solution if for each choice of values of the resources in the effects and in the initial state it is a classical solution (i.e. for deterministic domains).

In this work we have chosen to represent imprecision and vagueness on resource values in the preconditions and effects of actions with interval fuzzy sets. Therefore it is possible to represent an incomplete knowledge over resources, as in the initial state as in the action effects. We can handle cases like *"the fuel in the tank is surely between 10 and 15 but it would vary between 8 and 18"* or like *"we consume 0.1 l/Km with a possible error as* $\pm 0.015$ *l/km"*.

The solutions generated by planners which use this model present a higher degree of robustness, because they are executable in all the situations in which the resources values lie in the specified (fuzzy) intervals.

RDPPlan, the model of planning with resources which we will describe, is based on DPPlan [1], a planner which uses a non directional search algorithm on the planning graph. RDPPlan is compatible with the resources model as described in standard PDDL 2.1 [5], and extends it by allowing updates and assignments of quantities specified by fuzzy intervals.

In the following paragraphs, after recalling the main features about fuzzy interval and arithmetics on fuzzy intervals, we will present a model for planning

with fuzzy intervals. We will also present our prototype implementation of this model, the planner RDPPlan. This implementation of RDPPlan shows that the addition of vagueness resources management does not have a great impact on the overall DPPlan approach and on the planning graph structure. In the section about experimental results, we will show some performance of RDPPlan tested with planning domains from AIPS Planning Competition 2002.

## 2 An Introduction to Fuzzy Sets

The representation of non exact resources used in this paper is based on *fuzzy intervals*, a consistent and practical approach well established in fuzzy set theory [14]. In the following we will briefly recall the basic notions used in this work.

*Fuzzy intervals* are normal fuzzy sets defined on $\mathbb{R}$, whose $\alpha$-cuts are closed intervals of real numbers with bounded supports.

The membership function $\mu_A(x)$ of a fuzzy interval $A$ can be defined in canonical form by

$$\mu_A(x) = \begin{cases} f_A(x) & \text{if } x \in [a, b) \\ 1 & \text{if } x \in [b, c] \\ g_A(x) & \text{if } x \in (c, d] \\ 0 & \textit{otherwise} \end{cases} \tag{1}$$

where $a, b, c, d \in \mathbb{R}$, $a \leq b \leq c \leq d$, $f_A$ is a real-valued, increasing and right-continuous function, and $g_A$ is a real-valued, decreasing and left-continuous function. The $\alpha$-cut $A_\alpha$ of a fuzzy interval $A$, is defined by

$$A_\alpha = \begin{cases} [f_A^{-1}(\alpha), g_A^{-1}(\alpha)] & \text{if } \alpha \in (0, 1) \\ [b, c] & \alpha = 1 \end{cases} \tag{2}$$

In this paper, as in many fuzzy models, we can limit the discussion to *trapezoidal fuzzy intervals (TFIs)* (see Fig. 1) where $f_A$ and $g_A$ are linear functions. $TFIs$ can be represented in a computationally simple form and easily subsume other representations for uncertain resources such as triangular fuzzy numbers, crisp intervals and obviously crisp numbers.
Assuming $TFIs$, (1) and (2) can be written in the simpler forms

$$\mu_A(x) = \begin{cases} \frac{x-a}{b-a} & \text{if } x \in [a, b) \\ 1 & \text{if } x \in [b, c] \\ \frac{d-x}{d-c} & \text{if } x \in (c, d] \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

$$A_\alpha = [a + (b - a)\alpha, d - (d - c)\alpha], \quad \forall \alpha \in (0, 1]. \tag{4}$$

In the following we will use the shorthand symbol $A = (a, b, c, d)$ to represent the fuzzy interval having (3) as membership function.

Fig.1 points out the difference between a general fuzzy interval membership function and a trapezoidal one.

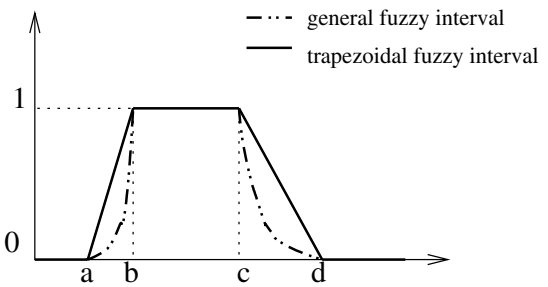



**Fig. 1.** Membership functions

### 2.1 Fuzzy Arithmetic

Arithmetic operations on fuzzy intervals can be defined using the extension principle of fuzzy set theory [7] [11]. Given an operation $\circ$, according to the extension principle, we have the general formula

$$\mu_{(A \circ B)}(c) = \sup_{c=a \circ b} \min\{A(a), B(b)\}. \tag{5}$$

For example, in the case of $TFI$ represented by a quadruple of real numbers, the formula for the operations $+$ (sum of intervals) and $-$ (difference of intervals) gives the following definitions

$$(a_1, a_2, a_3, a_4) + (b_1, b_2, b_3, b_4) = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4) \tag{6}$$

$$(a_1, a_2, a_3, a_4) - (b_1, b_2, b_3, b_4) = (a_1 - b_4, a_2 - b_3, a_3 - b_2, a_4 - b_1). \tag{7}$$

In order to manipulate fuzzy intervals it is also necessary to provide the *inclusion relationship*; the classical definition states that let $A$ and $B$ two fuzzy sets,

$$B \subseteq A \quad \text{iff} \quad \mu_A(x) \geq \mu_B(x), \quad \forall x \in A. \tag{8}$$

For the case of two $TFIs$ $A = (a_1, a_2, a_3, a_4)$ and $B = (b_1, b_2, b_3, b_4)$ (8) reduces to the simple linear conditions

$$\begin{cases} b_1 \geq a_1 \\ b_2 \geq a_2 \\ b_3 \leq a_3 \\ b_4 \leq a_4 \end{cases} \tag{9}$$

It is worth noticing that the $TFIs$ approach offers the advantage of a simple representation and the ability of defining interval operations which are very easy to compute.

## 3 A Model for Planning Resources with Fuzzy Intervals

Different approaches have been proposed in order to extend classical planner with the ability of manipulate resources [8] and different forms of uncertainty [12]. Fuzzy intervals allow to model adequately the imprecision of most resources managed in planning domains. The levels of typical planning resources (such as power supply, fuel, money etc.) are varying over intervals in a fairly continuous way, while they are produced or consumed by the execution of planning actions. In real situations the quantities which are actually produced/consumed cannot be precisely determined (e.g. fuel consumption per km) with a few exceptions (e.g. completely filling a tank), but they can be realistically restricted to an interval (e.g. consuming bewteen 3 and 3.5 liters of fuel per km). $TFIs$ are an effective way to represent resources bounded in an interval with fuzzy borders and, moreover, offer computational advantages with respect to a general fuzzy interval approach with generic and possibly computationally hard membership function.

The planning model for fuzzy resources is an extention of the classical planning model [5] characterized by the tuple $(I, G, O)$, where $I$ is the initial state, $G$ is a set of goal and $O$ is a set of operators described in term of preconditions/effects.

In our framework we provide the same tuple $(I, G, O)_f$ where each element is appropriately extended in order to handle fuzzy resources in preconditions, effects, initial state and goals. Conforming to PDDL2.1 [5], resources are represented as a numerical functions whose parameters can be domain constants or action parameters.

### 3.1 Initial State and Action Effects

The state of resource $r$ at time $t$ is denoted by $Val(r, t)$ and represented by a $TFI$:

$$Val(r, t) = (a_1, a_2, a_3, a_4) \tag{10}$$

The *initial state* of a resource is defined by $TFI$ assigned at time $t = 0$. The *effects* of an action executed at time $t$ can modify the state of a resource, at time $t + 1$, by *adding*, *subtracting* or *assigning* a $TFI$ to it. The semantics of the first two operations is the one given in (6), while the semantics of assignment is straightforward. The $TFIs$ to add/substract/assign are, in general, functions of the domain constants and the action parameters.

### 3.2 Preconditions and Goals

A *resource precondition* of an action can be specified, in the general form, as a constraint like

$$Val(r, t) \subseteq C \tag{11}$$

where $r$ is the resource name and $C$ is a $TFI$; the precondition is verified when condition (9) holds at time $t$.

A *resource goal* for $r$ specifies a resource constraint like (11) where $t$ is the final state of the solution plan.

It is worth noticing that the general form of constraint (11) allows to specify a wide range of constraints like *greater_than*, *less_than*, *between* etc.; this is made possible by the use of unlimited or degenerate $TFI\ C$.

### 3.3 Solution Plans and Action Executability

The complete semantics for action execution in this model follows directly as an extension of classical, purely logical planning.

An action is *executable* in a state $t$ if all its logical and resource preconditions hold in state $t$.

The state $S_{t+1}$ after the action execution is determined by the logical effects and by updating resources (add/subtract/assign) according to action effects.

A plan $P$ is a *solution plan* for a fuzzy resources domain $(I, G, O)_f$ if the logical/resource goals in $G$ hold in the final state resulting from executing $P$ in $I$.

## 4 RDPPlan: Planning with Uncertain Resources

In this section we introduce RDPPlan, a prototype planner for handling imprecise resources that implements the *TFIs model* presented in Section 3. RDPPlan has been derived from DPPlan [1], a purely logical planner based on planning-graph propagation rules. Initially we describe the syntax used for initial state, goals and action descriptions, then we introduce the *realized interval* and *desired interval*, the two main support structures used by RDPPlan algorithm, and their properties with respect to DPPLAN propagation rules are also given. Finally we describe how the classical notions of *simultaneous actions* execution and *solution plan* are adapted to this implementation framework.

### 4.1 Initial State and Action Effects

In the initial state the resources are initialized by the proposition `(= resource value)` where `value` can be a numerical value, a crisp interval, or, more in general, a $TFI$.

Action effects can change a resource by proposition like `(change resource value)`, where `change` can be one of the operators $=$, $+=$, $-=$ (respectively for assignemnt, decrement, increment) and `value` is an expression as in the preconditions.

We indicate the effect of the action $A$ on the resource $r$ with the $TFI\ E_{Ar}$ and, if an action $A$ does not change a resource $r$, we treat $A$ as an operator with null effect, i.e. $E_{Ar} = (0, 0, 0, 0)$.

### 4.2 Preconditions and Goals

The general form of constraints allowed in RDDPlan preconditions and goals is `(compare resource value)` where `compare` is a comparison operator in the set $(>, <, \leq, \geq, ==)$, `resource` is the name of resource involved and `value` can be an expression involving action parameters, numerical constants and arithmetic operators.

Since we do not allow for disjunctive preconditions and goals, several constraints on the same resource $r$ reduce to a unique $TFI$, possibly empty, indicated with $P_{Ar}$ for the precondition of action $A$ and with $G_r$ for goals. These $TFIs$ are possibly unlimited in left or in right side.

### 4.3 Realized Interval and Desired Interval

RDPPlan associates to each resource $r$ at each time level $t$, two $TFIs$ $R_{rt}$ and $D_{rt}$, which are computed and updated during the planning phase. The interval $R_{rt}$, called *realized interval*, represents the current value of the resource $r$ at time $t$ realized by the actions since now inserted in the plan. The interval $D_{rt}$, called *desired interval*, contains all the admissible values for the resource $r$ that allow the execution of all the actions selected at time level $t$.

Every time an action is selected at time level $\tau$ by the search procedure, $R_{rt}$ and $D_{rt}$ are updated for every resource $r$ used by the action and for every time level $t \geq \tau$. The previous intervals $R_{rt}$ and $D_{rt}$ are restored during the backtracking phase. In the following we denote by $T$ the last time level, i.e. the goal time level.

*Setting and propagating $R_{rt}$:*

At the start of search procedure, for each resource $r$ and for every time $t < T$, $R_{rt} = Val(r, 0)$.

When a new action $A$ having effects over a resource $r$ is chosen for the solution plan at time level $\tau$, each $R_{rt}$ is updated according the following simple rules:

. if $A$ has an assign effect `(= r` $E_{Ar}$`)`, then $R_{r,\tau+1} = E_{Ar}$
. if $A$ has an increase effect `(+= r` $E_{Ar}$`)`, then $R_{r,\tau+1} = R_{r\tau} + E_{Ar}$
. if $A$ has a decrease effect `(-= r` $E_{Ar}$`)`, then $R_{r,\tau+1} = R_{r\tau} - E_{Ar}$

using the arithmetic on $TFIs$ introduced in section 2.1.

For time levels $t > \tau$, $R_{rt}$ will be updated taking into account the effects of actions already chosen in these levels; in particular the updating is done a each time level until the one that has an assignment selected.

*Setting and propagating $D_{rt}$:*

At the start of search procedure

. for each resource $r$ and for every time $t < T$, $D_{rt} = (-\infty, -\infty, +\infty, +\infty)$
. for each resource $r$, $D_{rT} = G_r$.

When a new action $A$ is chosen for the solution plan, if A has precondition involving a resource $r$, a new $D_{rt}$ is computed using the methods presented in details in the following of the section.

### 4.4 Solution Plans and Action Executability

**Definition 1** *(Executable Plan) The sufficient and necessary condition for a plan to be executable and to be a solution of the given planning problem is that*

$$R_{rt} \subseteq D_{rt}, \quad \forall r \quad \forall t. \tag{12}$$

Before describing the rules for updating $D_{rt}$, it must be defined when two or more actions are executable at the same time level. The same concept of simultaneous executability as expressed in [10] and in [12] has been extended to the case of fuzzy resources.

**Definition 2** *(Simultaneous actions) A set of actions $\{A_1, A_2, \ldots, A_m\}$ is simultaneously executable if for every permutation $\pi$ of the actions in the set, $A_{\pi(1)}, A_{\pi(2)}, \ldots, A_{\pi(m)}$*

- *$A_{\pi(1)}$ is executable in the current state, $A_{\pi(2)}$ is executable in the state after the execution of $A_{\pi(1)}$, $A_{\pi(3)}$ is executable in the state after the execution of $A_{\pi(1)}$ and then $A_{\pi(2)}$, and so on;*
- *the effect over the resources is the same.*

As a straightforward consequence of the second condition, an assignment on resource $r$ is not simultaneously executable with any action changing $r$.

The question remains open whether to allow an action changing $r$ to be simultaneous with any action having a precondition with respect to $r$. Our approach is to allow simultaneity whenever the change does not affect the executability.

**Proposition 1** *Two additive actions $A_1$ and $A_2$ are simultaneously executable (with respect to $r$) if, let be*
$P_{A_1,r} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4) \quad P_{A_2,r} = (\beta_1, \beta_2, \beta_3, \beta_4)$
$E_{A_1,r} = (e_1, e_2, e_3, e_4) \quad E_{A_2,r} = (f_1, f_2, f_3, f_4)$
*we have* $v_1 \leq v_2 \leq v_3 \leq v_4$, *where*
$v_i = \max\{\alpha_i, \beta_i, \alpha_i - f_i, \beta_i - e_i\}, \quad i = 1, 2$ *and*
$v_j = \min\{\alpha_j, \beta_j, \alpha_j - f_j, \beta_j - e_j\}, \quad j = 3, 4.$

*In the positive case we set $D_{rt} = (v_1, v_2, v_3, v_4)$ if actions $A_1$ and $A_2$ are the only actions to be executed at time $t$. Otherwise $A_1$ and $A_2$ are marked to be mutually exclusive.*

The generalization to the case of many additive (over the same resource $r$) actions $A_1, A_2, \ldots, A_m$ is somehow straightforward.

**Proposition 2** *Let be*
$P_{A_i r} = (e_{1i}, e_{2i}, e_{3i}, e_{4i})$ $E_{A_i r} = (f_{1i}, f_{2i}, f_{3i}, f_{4i})$, *for* $i = 1, \ldots, m$.
*We have that the action $A_i$'s are simultaneously executable (with respect to $r$) if* $v_1 \leq v_2 \leq v_3 \leq v_4$, *where*
$v_1 = \max\{\alpha_{1i}, \alpha_{1i} - {k_i}^{--} : i = 1, \ldots, n\}$, $v_2 = \max\{\alpha_{2i}, \alpha_{2i} - {k_i}^{-} : i = 1, \ldots, n\}$,
$v_3 = \min\{\alpha_{3i}, \alpha_{3i} - {k_i}^{+} : i = 1, \ldots, n\}$, $v_4 = \min\{\alpha_{4i}, \alpha_{4i} - {k_i}^{++} : i = 1, \ldots, n\}$,

*and* $\forall i = 1, \ldots, n$
$k_i{}^{++} = \sum_{j \neq i, e_{4j} > 0} e_{4j}$, $k_i{}^{+} = \sum_{j \neq i, e_{3j} > 0} e_{3j}$, $k_i{}^{--} = \sum_{j \neq i, e_{1j} < 0} e_{1j}$, $k_i{}^{-} = \sum_{j \neq i, e_{2j} < 0} e_{2j}$.

*In the positive case* $D_{rt} = (v_1, v_2, v_3, v_4)$ *if the actions* $A_i$*'s are the only actions to be executed at time* $t$.

The terms $v_1, v_2, v_3, v_4$ can be computed in linear time (in the number of resources and actions) by storing (and keeping updated) the values
$k_0{}^{--} = \sum_{e_{1j} < 0} e_{1j}$, $k_0{}^{-} = \sum_{e_{2j} < 0} e_{2j}$, $k_0{}^{+} = \sum_{e_{3j} > 0} e_{3j}$, $k_0{}^{++} = \sum_{e_{4j} > 0} e_{4j}$.

The case of several additive actions includes the case of simultaneous execution of additive actions on resource $r$ (possibly none) with other actions which do not change $r$. In the particular case where all the actions do not change $r$, $D_{rt}$ reduces to the intersection of all the precondition intervals.

A case not yet covered is the simultaneous execution of an assignment on $r$, say $A_1$, whose assigns to $r$ the $TFI$ $(e_1, e_2, e_3, e_4)$, with actions $A_2, \ldots, A_n$ which do not change $r$.

It is easy to see that $A_1, A_2, \ldots, A_n$ are simultaneous executable (with respect to $r$) if the intersection of all the precondition intervals is not empty and contains $(e_1, e_2, e_3, e_4)$.

After a new action $A$ is selected at time $t$, we must check if it is simultaneously executable with the already selected actions at time $t$, by computing for each resource $r$ the quantities $v_1(r)$, $v_2(r)$, $v_3(r)$ and $v_4(r)$. Only when each of these $TFIs$ is not empty, then the desired interval for $r$ is set to be $(v_1(r), v_2(r), v_3(r), v_4(r))$.

Moreover, an incremental way which takes only a constant time to be computed of updating $D_{rt}$ after the choice of a new operator $A$ is the following.

**Proposition 3** *If* $D_{rt} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, $P_{Ar} = (a_1, a_2, a_3, a_4)$ *and* $E_{Ar} = (e_1, e_2, e_3, e_4)$, *then the updated desired interval is* $D'_{rt} = (\alpha'_1, \alpha'_2, \alpha'_3, \alpha'_4)$, *where*
$\alpha'_1 = \max\{a_1 - k_0{}^{--}, \alpha_1 - \min\{e_1, 0\}\}$, $\alpha'_2 = \max\{a_2 - k_0{}^{-}, \alpha_2 - \min\{e_2, 0\}\}$,
$\alpha'_3 = \min\{a_3 - k_0{}^{+}, \alpha_3 - \max\{e_3, 0\}\}$, $\alpha'_4 = \min\{a_4 - k_0{}^{++}, \alpha_4 - \max\{e_4, 0\}\}$.

Updating the values $R_{rt}$ is done by recomputing them for every resource changed by $A$, starting from time $t + 1$ and ending at the first time where an assignment over $r$ is selected. This computation can be efficiently done by storing and keeping updated the total amount to be added to $r$, computed considering all the actions selected since now.

## 5 Strategy for Resources

In this section we present the strategy that we have defined for achieving the goals over resources. This strategy is necessary to solve "pure numerical problems", i.e. problems with goals only on resources.

This strategy work exactly as the other strategies of DPPlan [1]: it chooses the best action to try first. But since backtracking is used when a contradiction is found, all the possible choices can be explored and the planner is complete.

The method that implements this strategy is combined with the ones for solving logical goals, by evaluating the difficulties of resources logical goals and selecting the most difficult goal to solve.

For each time step $t$ and for each resource $r$, we check if the condition $R_{rt} \subseteq D_{rt}$ holds: the negative cases are the goals on resources.

When we work with resources as $TFIs$ using the algebra explained in section 2.1, we have to handle with intervals whose width can in general only grow, except when an assignment is performed. In fact it is obvious that the operations defined with (6) produce $TFIs$ which have widths larger than the original ones. If you think that the width of realized interval represents, in some sense, the indetermination on resource value, we have that the larger the interval width, the larger the indetermination. Moreover note that if the width of the realized interval is large, it is more difficult that the solution condition (12) will hold. Let $|R|$ denote the greatest width of $TFI$ $R = (a_1, a_2, a_3, a_4)$, i.e. $|R| = a_4 - a_1$. The first control to do is on the widths of realized and desired intervals.

- If $|R_{rt}| > |D_{rt}|$ an assignment which assigns an interval with width less than $|D_{rt}|$ is the only possible choice. If there are many such assignments, the algorithm chooses that one which assigns the interval with the least width. If there are no assignments with this property, a backtracking is necessary.
- If $|R_{rt}| \leq |D_{rt}|$ the algorithm tries to solve this goal using this procedure:
  - the algorithm searches for an action that can achieve the goal in one only step preferring, in the case of many available options, the action situated at the level nearest to $t$. In this way it is unlikely that an action which can negate the goal is selected.
  - if this first search fails, the algorithm chooses an action that can permit us to come closer to the goal. The choice criterium, now, take into account the distance between $D_{rt}$ and $R'_{rt}$ and their widths.

This criterium has been implemented by defining two preference functions, one for the interval widths and one for the distance between intervals, and by searching for an action that maximizes a linear combination of the functions.

The first function is

$$f_W(A, D_{rt}) = \frac{|D_{rt}| - |R'_{rt}|}{|D_{rt}|} \tag{13}$$

and gives to each action $A$ a numerical positive score between 0 and 1.

The second function is a decreasing function of the distance between the middle points of the two intervals

$$f_D(A, D_{rt}) = \exp\left(-\frac{1}{2}\left|\sum_{\alpha=1}^{4} \alpha_i - \sum_{a=1}^{4} a_i\right|\right) \tag{14}$$

**Table 1.** Results on domain Zeno

| Zeno | |
|---|---|
| 2-4 | 2-6 |
| 0.07 secs | 29.55 secs |
| 11 acts | 21 acts |
| 5 lev. | 6 lev. |

**Table 2.** Results on domain Driver-Log

| DriverLog | | | |
|---|---|---|---|
| 2-2-4 | 3-2-4 | 3-2-5 | 3-3-5 |
| 0.26 secs | 164.30 secs | 400.67 secs | 0.83 secs |
| 12 acts | 18 acts | 19 acts | 11 acts |
| 7 lev. | 7 lev. | 8 lev. | 5 lev. |

**Table 3.** Results on Depots

| Depots | | |
|---|---|---|
| 1818 | 7512 | 1234 |
| 0.03 secs | 0.64 secs | 17.56 secs |
| 12 acts | 16 acts | 24 acts |
| 5 lev. | 8 lev. | 10 lev. |

where $(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = R'_{rt}$ and $(a_1, a_2, a_3, a_4) = D_{rt}$. Also this function gives to each action $A$ a numerical positive score between 0 and 1. The first experimental results show that good values for the ratio $c_1/c_2$ of the coefficients of the linear combination

$$f(A, D_{rt}) = c_1 f_W(A, D_{rt}) + c_2 f_D(A, D_{rt}) \tag{15}$$

are between 1 and 2. A theoretical justification is that $f_W$ is slightly more influent for driving the search algorithm, because it can be useless to get closer to the desired interval if the width of the realized interval is too large.

## 6 Experimental Results

Experiments have been done by testing the prototype RDPPlan on a significative number of problem and domains from the AIPS-2002 Planning Competition written in PDDL 2.1 [5]. It must be noted that although PDDL 2.1 models actions with resources, we had to modify these domains in order to describe domains with fuzzy resources. The three tables show the results obtained respectively with the Zeno, DriverLog and Depots domains. The rows shows respectively do-

main name, problem name, search time, number of actions and number of time level of the solution

Since there are no planners able to handle fuzzy resources and other planners can only use a limited sort of uncertainty on the resources [12], a real comparison cannot be performed. On the other hand, although an extensive experimentation is needed, preliminary results seem to suggest that RDPPlan performance does not decrease with respect to non fuzzy planner.

## 7 Conclusion

This work represents, to the best of our knowledge, the first plannner that can handle resources using a fuzzy approach. The semantics of the presented planning model makes extensive use of the notion of trapezoidal fuzzy intervals as a computationally feasible representation which adequately applies to most resources in planning domains.

The fuzzy model has been realised and experimented by RDPPlan, a fuzzy resources planner, which extends DPPlan a pure logical planner based on propagation rules. A notion of simultaneous actions execution for fuzzy resources and a strategy for searching with resource goals has been introduced for RDPPlan. An original contribution is that the strategy allow to search for mixed logical/quantitative goals and to solve problems with only pure quantitative resource goals.

An interesting future issue is represented by the definition of new strategies for planning in presence of fuzzy resources, and by the investigation of methods for comparing different approaches to planning with uncertainty.

## References

[1] M Baioletti, S Marcugini, and A Milani. DPPlan: an algorithm for fast solution extraction from a planning graph. In *Proc. of AIPS-2000*, 2000. 337, 341, 345

[2] S Chien, G Rabideau, R Knight, R Sherwood, B Engelhardt, D Mutz, T Estlin, B Smith, F Fisher, T Barret, G Stebbings, and D tran. ASPEN automated planning and scheduling for space mission operation. In *Proc. of SpaceOps 2000*, 2000. 336, 337

[3] K Currie and A Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, (52):49–86, 1991. 336

[4] M Do and S Kambhampati. Sapa: A domain independent heuristic metric temporal planner. In *Proc. of ECP-2001*, 2001. 337

[5] M Fox and D Long. PDDL 2.1: An extension to PDDL for expressing temporal planning domains. 2002. 337, 340, 346

[6] J Hoffmann. Extending FF to numerical state variables. In *Proc. of ECAI 2002*, 2002. 337

[7] A Kaufmann and M M Gupta. Introduction to fuzzy arithmmetic: Theory and applications. *Van Nostrand, New York*, 1985. 339

[8] J Koehler. Planning under resource constraints. In *Proc. of ECAI-1998*, 1998. 336, 340

[9] P Laborie and M Ghallab. Planning with sharable resource constraints. In *Proc. of IJCAI 1995*, 1995. 337

[10] D Long and M Fox. Efficient implementation of the plan graph in STAN. *Journal of Artificial Intelligence Research*, (10):87–115, 1999. 343

[11] M Mizumoto and K Tanaka. Some properties on fuzzy numbers. *Advances in Fuzzy Set Theory and Applications*, pages 153–164, 1979. 339

[12] J Rintanen and H Jungholt. Numeric state variables in constraint-based planning. In *Proc. of ECP-1999*, 1999. 336, 340, 343, 347

[13] S Wolfman and D Weld. The LPSAT engine and its application to resource planning. In *Proc. of IJCAI 1999*, 1999. 336

[14] L A Zadeh. Fuzzy sets. *Information and Control*, (8):338–353, 1965. 338

# Planning by Abstraction Using HW[]

Giuliano Armano, Giancarlo Cherchi, and Eloisa Vargiu

Department of Electrical and Electronical Engineering, University of Cagliari
I-09123, Cagliari, Italy
{armano,cherchi,vargiu}@diee.unica.it

**Abstract.** This paper presents the parametric system *HW[]*, devised and implemented to perform planning by abstraction. The parameter is an external PDDL-compliant planner, which is exploited to search for solutions at any required level of abstraction including the ground one. To represent abstraction hierarchies an extension to the standard PDDL notation has been devised, which allows one to control communication between adjacent levels by specifying suitable translation rules. Moreover, a novel semi-automatic technique for generating abstract spaces is also described. Five domains –taken from the AIPS 2002, 2000 and 1998 planning competitions– have been selected as test bed. To assess the impact of abstraction techniques on search performances, experiments have been focused on comparing three relevant planners with their hierarchical counterparts. Results put into evidence that the hierarchical approach may significantly speed up the search.

## 1 Introduction

Abstraction has been used to reduce search in a variety of planning systems (e.g.: GPS [1], ABSTRIPS [2], ABTWEAK [3], PABLO [4], PRODIGY [5]). Common experimental results have shown that abstraction is usually not effective on simple problems, due to the overhead introduced by the need of going back and forth across abstract spaces while performing the search. In other words, enforcing abstraction on simple problems may end up to a waste of computational resources. Yet, under certain assumptions abstraction can reduce the size of the search space from exponential to linear in the size of the solution [6].

So far, existing planning systems have been focusing on improving a specific algorithm without taking into account the possibility of generalizing the approach to a generic algorithm. To investigate on the possibility of improving the performances of a generic planner, we devised and implemented a hierarchical wrapper able to embody any domain-independent planner provided that a compliance with the STRIPS subset of PDDL 1.2 standard [7] is ensured. The embodied planner is exploited at any level of the hierarchy, each level being characterized by its own definitions. A suitable decoupling between levels is guaranteed by using domain-specific rules that describe how to translate the representation language from a level to its superior and vice-versa [8]. To express this kind of translation rules, an extension to PDDL standard notation has been adopted, explicitly defined to support abstraction hierarchies [9].

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 349-361, 2003.
© Springer-Verlag Berlin Heidelberg 2003

Experiments have been performed by embedding into the system three different planners. Some classical planning domains, widely acknowledged by the planning community, have been used as test beds. The remainder of this paper is organized as follows: first, some relevant work on planning by abstraction is briefly recalled, to give the reader a better insight of the issues deemed relevant. Then, the overall architecture of the system is illustrated, including a semi-automatic technique developed for generating abstract spaces, and the adopted extension to the PDDL notation. Subsequently, experiments are described and results are discussed. Finally, conclusions are drawn and future work is outlined.

## 2 Related Work

Building an ordered set of abstractions for controlling the search has proven to be an effective approach for dealing with the complexity of planning tasks. This technique requires the original search space to be mapped into corresponding abstract spaces, in which irrelevant details are disregarded at different levels of granularity.

Let us briefly recall some relevant techniques proposed in the literature: (i) action-based, (ii) state-based, (iii) case-based, and (iv) Hierarchical Task Networks.

The first combines a group of actions to form macro-operators [10]. The second exploits representations of the world given at a lower level of detail; its most significant forms rely on (a) relaxed models [2], obtained by dropping operators' applicability conditions, and on (b) reduced models [11], obtained by completely removing certain conditions from the problem space. Both models, while preserving the provability of plans that hold at the ground level, perform a “weakening” of the original problem space, thus suffering from the drawback of introducing “false” (i.e., not refinable) solutions at the abstract levels [12]. In the third [13], abstract planning cases are automatically learnt from given concrete cases, as done in the PARIS system, although the user must provide explicit refinement rules between adjacent levels in the hierarchy. In the fourth [14], problem and operators are organized into a set of tasks. A high-level task can be reduced to a set of ordered lower-level tasks, and a task can be reduced in several ways. Reductions allow specifying how to obtain a detailed plan from an abstract one.

## 3 HW[] Architecture

*HW[]* stands for (parametric) Hierarchical Wrapper. Note that square brackets are part of the name, pointing to the ability to embed an external planner. Being *P* any such planner, the notation *HW[P]* shall be used to denote an instance of *HW[]* able to exploit the planning capabilities of *P*. The system can embed any domain-independent planner, provided that a compliance with the STRIPS subset of the PDDL 1.2 standard is ensured.

In principle, each level of abstraction may contain a different planner, thus permitting to select the most suitable planner for each level. [1] Nevertheless, in this

[1] In this case, a natural notation for highlighting the ordered set of embedded planners would be $HW[P_0, P_1, \ldots P_n]$, $P_i$ being the planner embedded at the $i^{th}$ level of the hierarchy.

paper, we assume that the same external planner is used at each level of abstraction. Moreover, we assume that only one abstract level exists –giving rise to a two-levels (i.e., ground and abstract) hierarchical description.

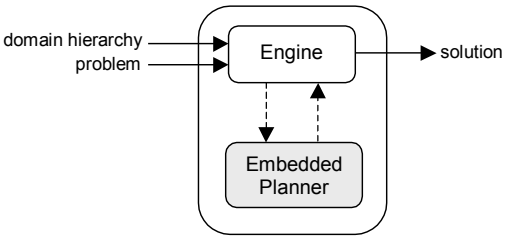



**Fig. 1.** HW[] architecture

Figure 1 sketches the architecture of the system, focusing on its two main components, i.e., an engine and the embedded planner. The former controls the communication between adjacent levels, whereas the latter performs planning at any given level of abstraction.

### 3.1 The Planning Algorithm

Once instantiated with an external planner *P*, *HW[P]* takes as inputs a ground-level problem and a structured description of the corresponding domain, including a set of rules to be used while mapping ground into abstract states and vice-versa. In fact, to perform planning at different levels of abstraction, the engine of *HW[]* must operate bi-directional translations (upwards and downwards) to permit communication between adjacent levels.

To find a solution of a given problem, first the engine of *HW[P]* translates the init and goal sections from the ground to the abstract level. *P* is then invoked to search for an abstract solution. Subsequently, each abstract operator is refined by repeatedly invoking *P*. The refinement of an abstract operator is performed by activating *P*, at the ground level, on the goal obtained by translating downward its effects. Note that the initial state of each refinement depends on the previous refinement; hence, refinements must be performed according to the order specified by the abstract plan. To avoid incidental deletion of subgoals already attained during previous refinements, they are added to the list of subgoals that results from translating downward the effects of the current abstract operator to be refined. When the attempt to refine the current abstract solution fails, *P* is invoked to find the next abstract solution [2] unless the number of abstract solutions found so far exceeds a given threshold. If no abstract solution could be successfully refined, to ensure the completeness of the algorithm an overall search is performed at the ground level. The whole process ends when a ground solution is found or the overall search fails.

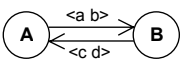
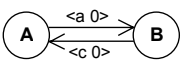
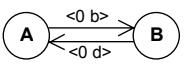
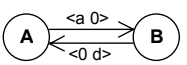
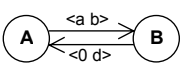
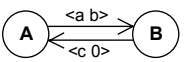
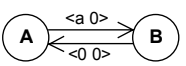
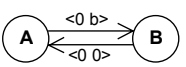
[2] Due to the limitations of most of the existing planners, the process of incrementally querying for another solution may be simulated by preliminarily querying for *m* abstract solutions to be released incrementally on demand.

### 3.2 Generating Abstractions

To facilitate the setting of abstract spaces, as an alternative to the hand-coded approach used in [15], a novel semi-automatic technique for generating abstraction hierarchies starting from ground-level domain descriptions has been devised and adopted.

From our particular perspective, performing abstraction basically involves executing two steps: (i) searching for macro-operator schemata through a priori or a posteriori analysis, (ii) selecting some of the schemata evidenced so far and translating them into abstract operators.

**Table 1.** Heuristics for pruning the operators' graph

| Type | Node relationships | Supporting Evidence | Action |
|---|---|---|---|
| 1 |  | (i) if a = c and b = d there is no supporting evidence for assuming that A usually precedes B in a plan, and vice-versa; | remove both edges. |
| | | (ii) if a > c there is a high likelihood that A precedes B; | remove top edge. |
| | | (iii) if c > a, there is a high likelihood that B precedes A. | remove bottom edge. |
| 2 |  | (i) if a > c there is a high likelihood that A precedes B; | remove top edge. |
| | | (ii) if c > a, there is a high likelihood that B precedes A. | remove bottom edge. |
| 3 |  | A (B) negates one or more preconditions required by B (A). | remove both edges. |
| 4 |  | B negates one or more preconditions required by A. | remove bottom edge. |
| 5 |  | B negates one or more preconditions required by A. | remove both edges. |
| 6 |  | A and B are usually complementary or loosely-coupled actions. | remove both edges. |
| 7 |  | A precedes B with high likelihood. | remove top edge. |
| 8 |  | A negates one or more preconditions required by B. | remove top edge. |

In this subsection, we concentrate on the task of finding macro-operator schemata throughout an *a-priori* analysis performed on the given domain and problem, rather than adopting the *a-posteriori* technique illustrated in [16], headed at finding macro-operator schemata according to a "post-mortem" analysis performed on plan "chunks".

Step (i) is performed by an algorithm for building and then pruning a directed graph, whose nodes represent operators and whose edges represent relations between effects of the source node and preconditions of the destination node. In particular, for each source node *A* and for each destination node *B*, representing operators defined in the given domain, the corresponding edge is labeled with a pair of non-negative numbers, denoted by *<a b>*. The pair accounts for how many predicates *A* can establish (*a*) and negate (*b*) that are also preconditions of *B*. It is worth noting that source and destination node may coincide, thus giving rise to a self-reference.

Pruning is performed according to the domain-independent heuristics reported in Table 1. Note that the pruned graph does not contain edges labeled *<0 0>*, the corresponding operators being completely independent.

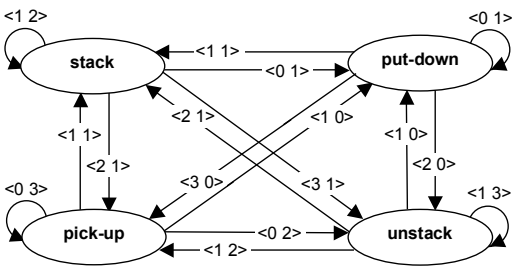



**Fig. 2.** The directed graph (before pruning), representing static relations between operators of the blocks-world domain

As an example, let us consider the well-known blocks-world domain, encompassing four operators: *stack*, *pick-up*, *unstack*, *put-down*. The corresponding graph is shown in Figure 2. Bearing in mind that the same mechanism has been applied to all operators' pairs, let us concentrate –for instance– on the relation that holds between *stack* (source node) and *pick-up* (destination node). Considering that the effects of the *stack* operator are:

```
(not (holding ?x))
(not (clear ?y))
(clear ?x)
(handempty)
(on ?x ?y)
```

and that the preconditions of the *pick-up* operator are:

```
(clear ?x)
(ontable ?x)
(handempty)
```

we label the corresponding edge with the pair *<2 1>*. It is apparent that stack establishes two preconditions for *pick-up*, while negating another.

As for the pruning activity, Figure 3 shows the resulting graph for the *blocks-world* domain.[3] The resulting macro-operator schemata are (";" being used for concatenation): *pick-up;stack*, *unstack;put-down*, *pick-up;put-down*, and *stack;unstack*.

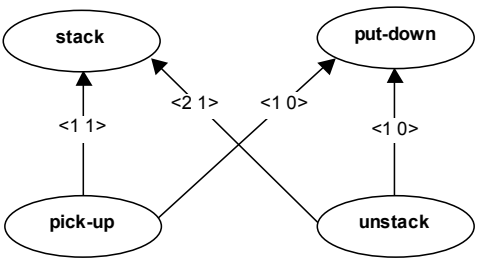



**Fig. 3.** The directed graph (after pruning), representing static relations between operators of the blocks-world domain

Step (ii) is performed by selecting the most promising macro-operator schemata from the pruned graph and by translating them into abstract operators. In the current implementation of the system, this step has not yet been completely automated. Nevertheless, the simplest way of generating an abstract operator consists of "simplifying" a macro-operator schema, say σ, expressed as a sequence of ground operators ω (i.e., $\sigma = \omega_1\omega_2....\omega_n = \sigma_1\omega_n$), whose preconditions (γ) and effects[4] (η) are defined as:

$$\begin{cases} \gamma_\sigma = \gamma_{\sigma_1} \cup \left(\gamma_{\omega_n} \setminus \eta_{\sigma_1}\right) \\ \alpha_\sigma = \left(\alpha_{\sigma_1} \setminus \delta_{\sigma_1}\right) \cup \alpha_{\omega_n} \\ \delta_\sigma = \left(\delta_{\sigma_1} \setminus \alpha_{\sigma_1}\right) \cup \delta_{\omega_n} \end{cases}$$

To perform simplification, all predicates that do not occur among preconditions or effects of any macro-operator obtained by inspecting the pruned graph should be deleted from the abstract level. This process influences (and is influenced by) the translation rules that apply to both types and predicates. For instance, if a predicate do not appear –as a precondition or effect– in any induced abstract operator, it can be deleted from the abstract level.

In particular, recalling the *blocks-world* example, *stack;unstack* and *pick-up;put-down* have been disregarded, being apparent that do not alter the state of the domain – i.e., the resulting set of effects is empty. Hence, the selected macro-operator schemata are *pick-up;stack* and *unstack;put-down* (see Table 2). As for the generation of abstract operators, let us stress that the predicate *(holding ?x)* does not appear in any precondition or effect; thus it can be removed by the abstract space. Furthermore, also

[3] Since we are interested in finding macro-operators, we do not take into account self-references.

[4] Let us recall that effects can be split into add-list (α) and delete list (δ), representing positive and negative effects, respectively.

*(handempty)* has been removed as there is no operator (at the abstract level) able to negate it.

**Table 2.** Selected macro-operator schemata

| Macro-Operator Schema | Ground Sequence | Preconditions | Effects |
|---|---|---|---|
| (pick-up&stack ?b1 ?b2 - block) | pick-up; stack | (clear ?b1) (ontable ?b1) (handempty) (clear ?b2) | (not (ontable ?b1)) (not (clear ?b2)) (on ?b1 ?b2) |
| (unstack&put-down ?b1 ?b2 - block) | unstack; put-down | (clear ?b1) (on ?b1 ?b2) (handempty) | (ontable ?b1) (clear ?b2) (not (on ?b1 ?b2)) |

As a final comment, let us worth pointing out that the approach described above can be used also for generating abstractions tailored to a given problem; this can be done by simply adding a dummy operator representing the goal(s) of the problem itself. The "goal" operator has only preconditions (its set of effects being empty), representing a logic conjunct of predicates that characterize the goal of the input problem. As a consequence, all sequences deemed relevant to solve the problem are easily put into evidence (as they end with the "goal" operator).

### 3.3 Extending PDDL for Dealing with Abstraction

A problem and its corresponding domain are described in accordance with the standard PDDL 1.2 syntax, using the "*define problem*" and "*define domain*" statements, respectively. To describe how bi-directional communication occurs between adjacent levels an extension to PDDL standard has been devised and adopted.

More precisely, the syntactic construct *define hierarchy* has been introduced. It encapsulates an ordered set of domains, together with a corresponding set of mappings between adjacent levels of abstraction. Since the mappings are given in term of types, predicates and operators, three subfields have been defined (i.e. `:types`, `:predicates`, and `:actions`), to represent the abstraction over such dimensions. The general form of the construct is:

```
(define (hierarchy <name>)
  (:domains <domain-name>+)
  (:mapping  (<source-domain> <destination-domain>)
   [:types <types-def>]
   [:predicates <predicates-def>]
   [:actions <actions-def>])*)
```

To represent how types are mapped between adjacent levels, in the `:types` field a list of clauses in the following notation must be given:

```
(abstract-type ground-type)
```

It specifies that ground-type becomes abstract type while performing upward translations. To disregard a ground-type, the following notation must be used:

```
(nil ground-type)
```

Moreover, to represent how predicates are mapped between adjacent levels, in the `:predicates` field a list of clauses in the following notation must be given:

```
((abstract-predicate ?p11 ?p21 …)
 (ground-predicate   ?p12 - t12 ?p22 - t22 …))
```

It specifies that the ground-predicate must be preserved while going upward and vice-versa. [5]

To disregard a predicate while performing upward translations, the following notation is be used:

```
(nil (ground-predicate  ?p12 - t12 ?p22 - t22 …))
```

It specifies that ground-predicate is not translated into any abstract-level predicate.

In addition, abstract-predicate can be expressed as a logical combination of some ground level predicates.

```
(define (hierarchy blocks)
  (:domains blocks-ground blocks-abstract)
  (:mapping
    (blocks-ground blocks-abstract)
    :predicates
      ((nil (handempty))
       (nil (holding ?b - block)))
     :actions
      ((nil (pick-up ?b))
       (nil (put-down ?b))
       (nil (stack ?b1 ?b2))
       (nil (unstack ?b1 ?b2))
       ((pick-up&stack ?b1 ?b2)(and (pick-up ?b1) (stack ?b1 ?b2)))
       ((unstack&put-down ?b1 ?b2)(and (unstack ?b1 ?b2) (put-down ?b1))))))
```

**Fig. 4.** Hierarchy definition of the blocks-world domain

To describe how to build the set of operators for the abstract domain, in the `:actions` field four kind of mapping can be expressed:

1. An action is removed;
2. An action is expressed as a combination of ground domain actions;
3. An action remains unchanged or some of its parameters are disregarded;
4. A new operator is defined from scratch.

[5] If no differences exist in mapping a predicate between adjacent levels the corresponding clause can be omitted.

**Table 3.** Performance comparison of *BB*, *GP*, and *LPG* together with their hierarchical counterparts *HW[BB]*, *HW[GP]*, *HW[LPG]*

| # | GP | HW[GP] | BB | HW[BB] | LPG | HW[LPG] |
|---|---|---|---|---|---|---|
| *elevator* | | | | | | |
| 1-4 | 0.01 | 0.06 | 0.1 | 0.33 | 0.01 | 0.11 |
| 3-1 | 0.23 | 0.36 | 1.34 | 1.20 | 0.02 | 0.15 |
| 4-1 | 1.96 | 0.83 | 1.03 | 1.74 | 0.02 | 0.16 |
| 4-4 | 10.11 | 0.84 | 311.5 | 1.79 | 0.02 | 0.16 |
| 5-1 | 364.7 | 2.03 | 180.8 | 2.54 | 0.02 | 0.18 |
| 7-2 | -- | 12.04 | -- | 3.89 | 0.03 | 0.29 |
| *logistics* | | | | | | |
| 4-2 | 0.68 | 1.22 | 0.27 | 0.46 | 17.93 | -- |
| 5-2 | 0.08 | 0.16 | 0.15 | 0.46 | 0.02 | -- |
| 7-0 | -- | 10.93 | 4.49 | 2.17 | 2.12 | -- |
| 8-1 | -- | 16.26 | 2.90 | 3.02 | 1.55 | -- |
| 10-0 | -- | 43.43 | 8.27 | 3.76 | 2.17 | -- |
| 15-0 | -- | 203.4 | 10.91 | 6.33 | 0.15 | -- |
| *blocks-world* | | | | | | |
| 4-0 | 0.34 | 0.32 | 0.16 | 0.67 | 0.02 | 0.08 |
| 6-0 | 3.04 | 1.82 | 0.26 | 1.68 | 0.05 | 0.23 |
| 8-0 | 31.61 | 11.13 | 0.92 | 2.46 | 0.36 | 0.31 |
| 10-0 | -- | -- | 6.82 | 5.00 | 0.62 | 0.67 |
| 11-0 | -- | -- | 16.23 | 4.25 | 4.23 | 0.83 |
| 14-0 | -- | -- | -- | 9.84 | 5.00 | 1.91 |
| 15-0 | -- | -- | -- | -- | 7.49 | 2.07 |
| 17-0 | -- | -- | -- | -- | 33.93 | 3.49 |
| 20-0 | -- | -- | -- | -- | 66,78 | 7.88 |
| 22-0 | -- | -- | -- | -- | 183.16 | 12.21 |
| 25-0 | -- | -- | -- | -- | 668.98 | 24.94 |
| *zeno-travel* | | | | | | |
| 1 | 0.02 | 0.52 | 0.22 | 0.36 | 0.02 | 0.03 |
| 8 | -- | 42.55 | 0.94 | 2.36 | 0.14 | 0.49 |
| 9 | -- | -- | 0.34 | 3.37 | 0.13 | 1.08 |
| 11 | -- | -- | 11.20 | 2.78 | 0.16 | 1.06 |
| 13 | -- | -- | 62.99 | 20.52 | 0.42 | 2.47 |
| 14 | -- | -- | -- | 20.04 | 3.90 | 21.93 |
| *gripper* | | | | | | |
| 2 | 4.72 | 0.56 | 0.42 | 0.63 | 0.02 | 0.07 |
| 3 | 7.91 | 1.73 | 5.22 | 1.20 | 0.02 | 0.12 |
| 4 | 18.32 | 2.63 | 268.7 | 1.55 | 0.02 | 0.14 |
| 5 | 57.21 | 4.38 | 421.1 | 1.54 | 0.03 | 0.15 |
| 6 | -- | 7.97 | 586.4 | 2.26 | 0.03 | 0.17 |
| 9 | -- | 24.29 | -- | 3.63 | 0.05 | 0.36 |

To show how abstracting actions influences the whole hierarchy, let us recall the *blocks-world* domain. In this case the type hierarchy has not been abstracted, as it contains only one type (*block*). A typical abstraction based on macro-operators has been performed according to the results summarized in the previous subsection. The decision of adopting the operators *pick-up&stack* and *unstack&put-down* entails a deterministic choice on which predicates have to be forwarded / disregarded while performing upward translations. Figure 4 shows the corresponding hierarchical definition of the blocks-world domain, according to the proposed notation.

## 4 Experimental Results

The current prototype of the system has been implemented in C++. Experiments have been performed with three planners: *GRAPHPLAN* [17], *BLACKBOX* [18], and *LPG* [19]. In the following, *GP*, *BB*, and *LPG* shall be used to denote the *GRAPHPLAN*, *BLACKBOX*, and *LPG* algorithms, whereas *HW[GP]*, *HW[BB]*, and *HW[LPG]* shall be used to denote their hierarchical counterparts.

To assess the capability of abstraction to improve the search, we performed some tests on five domains taken from the 1998, 2000, and 2002 AIPS planning competitions [20], [21], [22]: *elevator*, *logistics*, *blocks-world*, *zeno-travel*, and *gripper*. Experiments were conducted on a machine powered by an Intel Celeron CPU, working at 1200 Mhz and equipped with 256Mb of RAM. A time bound of 1000 CPU seconds has also been adopted, and the threshold (m) used to limit the search for abstract solutions has been set to 1 for each planner.

All domains have been structured according to a ground and an abstract level; the latter having been generated following the approach described in the previous subsection. For each domain, several tests have been performed –characterized by increasing complexity. Table 3 compares the CPU time of each planner over the set of problems taken from the AIPS planning competitions. Dashes show problem instances that could not be solved by the corresponding system within the adopted time-bound.

*Elevator*. Experiments show that –for *GP* and *BB*– the CPU time increases very rapidly while trying to solve problems of increasing length, whereas *HW[GP]* and *HW[BB]* keep solving problems with greater regularity (although the relation between number of steps and CPU time remains exponential). *LPG* is able to solve long plans in a very short time, thus doing away the need to resort to *HW[LPG]*.

*Logistics*. In this domain *GP* easily solves problems up to a certain length but it is unable to solve problems within the imposed time limits if a given threshold is exceeded. On the other hand, *HW[GP]* keeps solving problems of increasing length without encountering the above difficulties. *BB* performs better than *HW[BB]* for small problems, whereas *HW[BB]* outperforms *BB* on more complex problems. *LPG* is able to solve long plans in a few seconds at the most. For unknown reasons *LPG* was not able to refine any abstract operator when invoked by the engine of *HW*.

*Blocks-world*. Tests performed on this domain reveal a similar trend for *GP* and *HW[GP]*, although the latter performs slightly better than the former. *BB* performs

better than *HW[BB]* for simple problems, whereas *HW[BB]* outperforms *BB* on problems of medium complexity. *LPG* is able to solve problems whose solution length is limited to 100 steps. In this domain, *HW[LPG]* clearly outperforms *LPG* on more complex problems.

*Zeno-travel*. Unfortunately, neither *GP* nor *HW[GP]* are able to successfully tackle any problem of this domain. An improvement of *HW[BB]* over *BB* can be observed, similar to the one shown for the *blocks-world* domain. *LPG* is able to solve long plans in a few seconds at the most, thus avoiding the need to resort to *HW[LPG]*.

*Gripper*. For the gripper domain, both *HW[GP]* and *HW[BB]* clearly outperform their non-hierarchical counterparts. *LPG* is able to solve long plans in a very short time.

## 5 Conclusions and Future Work

In this paper a parametric system has been presented, devised to perform planning by abstraction. The actual search is delegated to a set of external planners ("the parameter"). Aimed at giving a better insight of whether or not the exploitation of abstract spaces can be useful for solving complex planning problems, comparisons have been made between any instances of the hierarchical planner and its non-hierarchical counterpart.

To better investigate the significance of the results, three different planners have been used to make experiments. To facilitate the setting of experiments, a novel semi-automatic technique for generating abstract spaces has been devised and adopted. Experimental results highlight that abstraction is useful on classical planners, such as *GP* and *BB*. On the contrary, the usefulness of resorting to hierarchical planning for the latest-generation planner used for experiments (i.e., *LPG*) clearly emerges only in the *blocks-world* domain.

As for the future work, we are currently addressing the problem of automatically generating abstract operators.

## References

[1] Newell, A., Simon, H.A.: Human Problem Solving. Prentice Hall, Englewood Cliffs, NJ (1972)

[2] Sacerdoti, E.D.: Planning in a hierarchy of abstraction spaces. Artificial Intelligence, Vol. 5 (1974) 115-135

[3] Yang, Q., Tenenberg, J.: Abtweak: Abstracting a Nonlinear, Least Commitment Planner. Proceedings of the 8th National Conference on Artificial Intelligence, Boston, MA (1990) 204-209

[4] Christensen, J.: Automatic Abstraction in Planning. PhD thesis, Department of Computer Science, Standford University (1991)

[5] Carbonell, J., Knoblock, C.A., Minton, S. (ed.): PRODIGY: An integrated architecture for planning and learning. In D. Paul Benjamin Change of Representation and Inductive Bias. Kluwer Academic Publisher (1990) 125-146
[6] Knoblock, C.A.: Search Reduction in Hierarchical Problem Solving. Proceedings of the 9th National Conference on Artificial Intelligence, Anaheim, CA (1991) 686-691
[7] McDermott, D., Ghallab, M., Howe, A., Knoblock, C.A., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL – The Planning Domain Definition Language, Technical Report CVC TR-98-003 / DCS TR-1165, Yale Center for Communicational Vision and Control (1998)
[8] Armano, G., Cherchi, G., Vargiu, E.: A Parametric Hierarchical Planner for Experimenting Abstraction Techniques. Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03), Acapulco, Mexico (2003)
[9] Armano, G., Cherchi, G., Vargiu, E.: An Extension to PDDL for Hierarchical Planning. Workshop on PDDL (ICAPS'03), Trento, Italy (2003)
[10] Korf, R.E.: Planning as Search: A Quantitative Approach. Artificial Intelligence, Vol. 33(1) (1987) 65-88
[11] Knoblock, C.A.: Automatically Generating Abstractions for Planning. Artificial Intelligence, Vol. 68(2) (1994) 243-302
[12] Giunchiglia, F., Walsh, T.: A theory of Abstraction, Technical Report 9001-14, IRST, Trento, Italy (1990)
[13] Bergmann, R., Wilke, W.: Building and Refining Abstract Planning Cases by Change of Representation Language. Journal of Artificial Intelligence Research (JAIR), Vol. 3 (1995) 53-118
[14] Erol, K., Hendler, J., and Nau, D.S.: HTN Planning: Complexity and Expressivity. In Proceedings of the 12th National Conference on Artificial Intelligence (AAAI-94), AAAI Press / MIT Press, Seattle, WA (1994) 1123-1128
[15] Armano, G., Cherchi, G., Vargiu, E.: Experimenting the Performance of Abstraction Mechanisms through a Parametric Hierarchical Planner. Proceedings of IASTED International Conference on Artificial Intelligence and Applications (AIA'2003), Innsbruck, Austria (2003)
[16] Armano, G., Vargiu, E.: An Adaptive Approach for Planning in Dynamic Environments. Proceedings of the International Conference on Artificial Intelligence (IC-AI 2001), Special Session on Learning and Adapting in AI Planning, Las Vegas, Nevada (2001) 987-993
[17] Blum, A.L., Furst, M.L.: Fast Planning through Planning Graph Analysis. Artificial Intelligence, Vol. 90(1-2) (1997) 279-298
[18] Kautz, H., Selman, B.: BLACKBOX: A New Approach to the Application of Theo-rem Proving to Problem Solving. In Working notes of the Workshop on Planning as Combinatorial Search, AIPS-98, Pittsburg, PA (1998) 58-60
[19] Gerevini, A., Serina, I.: LPG: A Planner Based on Local Search for Planning Graphs. Proceedings of the 6th International Conference on AI Planning and Scheduling, AAAI Press, Menlo Park (2000)

[20] Long, D.: The AIPS-98 Planning Competition. AI Magazine, Vol. 21(2) (1998) 13-33
[21] Bacchus, F.: Results of the AIPS 2000 Planning Competition (2000) URL: http://www.cs.toronto.edu/aips2000
[22] Long, D.: Results of the AIPS 2002 Planning Competition (2002) URL: http://www.dur.ac.uk/d.p.long/competition.html

# The Role of Different Solvers in Planning and Scheduling Integration

Federico Pecora and Amedeo Cesta

Planning and Scheduling Team, Institute for Cognitive Science and Technology
Italian National Research Council
{pecora,cesta}@ip.rm.cnr.it
http://pst.ip.rm.cnr.it

**Abstract.** This paper attempts to analyze the issue of planning and scheduling integration from the point of view of *information sharing*. This concept is the basic bridging factor between the two realms of problem solving. In fact, the exchange of each solver's point of view on the problem to be solved allows for a synergetic effort in the process of searching the space of states. In this work, we show how different solving strategies cooperate in this process by varying the degree of integration of the combined procedure. In particular, the analysis exposes the advantage of propagating sets of partial plans rather than reasoning on sequential state space representations. Also, we show how this is beneficial both to a component-based approach (in which information sharing occurs only once) and to more interleaved forms of integration.

## 1 Introduction

The reason for the richness of research and debate in the area of planning and scheduling integration is that the solving techniques for planning and scheduling problems are fundamentally different. On one hand, a planner reasons in terms of causal dependencies: by performing logical deduction in a predefined logic theory, it is capable of deriving sets of actions which have to be performed in order to achieve a preset goal. On the other hand, scheduling involves the propagation of constraints, which in turn determines the domains in which the searched assignment of variables is to be found.

Both planning and scheduling are, basically, search algorithms. Planners and schedulers implement efficient techniques for (1) representing the search space and (2) finding solutions in the search space. Thus, research in planning and scheduling technology alike is directed towards, on one hand, finding data structures which represent the problems, and evolving the techniques with which these data structures are generated and explored on the other.

The goal of this paper is to assert some basic principles related to planning and scheduling integration. In particular, we formulate the idea of an integrated system in terms of *information sharing*, that is the level of synergy between the planning and the scheduling solving cores. These considerations will enable us to identify the solving algorithms that are best fit for planning and scheduling integration.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 362–374, 2003.
© Springer-Verlag Berlin Heidelberg 2003

## 2 Integrating Planning and Scheduling

Planning and scheduling address complementary aspects of problems, and this makes them a natural couple. As a consequence, research in the field of integrated planning and scheduling has recently produced some interesting results.

Many approaches to planning and scheduling integration have been proposed. Some deal with time and resources contextually with causal reasoning, that is they expand a strictly causal solving technique (such as GRAPHPLAN) with data structures capable of modeling time and/or resources. Some examples of this approach are given in [8, 9, 11, 17].

Another approach to integrating planning and scheduling capabilities in a single solving tool is to implement the two solving paradigms separately, and then to link them in order to solve the two aspects of the problem with two distinct solving techniques. This, which may seem the most realistic approach, presents some major difficulties since it requires the definition of forms of *information sharing* between the two subsystems. Indeed, in this paper we would like to assert the need for an in-depth comprehension of some fundamental aspects related to this form of integration of planning and scheduling. In particular, we will try to answer the following questions:

**When do we need an integrated reasoner?** This equates to identifying which sorts of problems require such combined reasoning capabilities. Thus, we will try to single out the characteristics of these problems in terms of whether they are fit for causal reasoning (what we have called planning) or time- and resource-related reasoning (scheduling).

**Which planning and scheduling technology is to be used?** This issue is clearly a very open one. The goal of this paper is to assert some fundamental principles of planning and scheduling integration. This involves the theorization of what we have called a Naïve Component-Based Approach (N-CBA), in which there is an explicit distinction between the two aspects of problems. We will see that the considerations which can be made in the N-CBA conform to those made in a more in-depth analysis of strongly interwoven planning and scheduling integrations.

In the following sections we will address the two questions contextually, by investigating the role of two solving ideas in the context of the N-CBA and of more tightly integrated approaches.

## 3 A Naïve Component-Based Approach

Probably the most immediate form of integration we can think of is to serialize a planner and a scheduler (hence the name Naïve Component-Based Approach, or N-CBA), as depicted in figure 1. A planner is given a problem specification along with a domain description *which is purified with respect to time- and resource-related constraints*. These constraints are accommodated after the planning procedure has taken place, i.e. has produced a Partially Ordered Plan

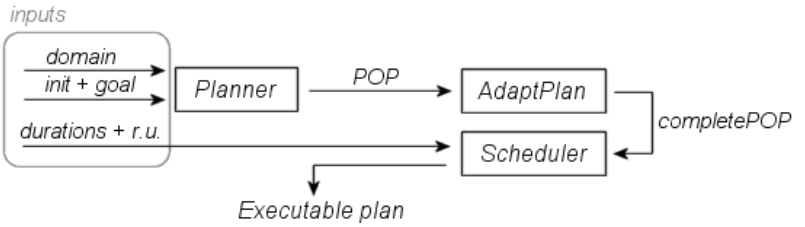



**Fig. 1.** Architecture of the N-CBA

(POP). The plan adaptation procedure is responsible for inserting this additional information. A scheduler is then employed in order to obtain a completely instantiated solution of the initial planning problem.

This approach is strongly intuitive, but we believe it is instrumental for the comprehension of the phenomena underlying the integration of the two processes. In fact, the *choice of components* for the serialized planning and scheduling system exposes very clearly the relative fitness of particular planning and scheduling solving strategies. In other words [1], studying both processes in a separate way has the effect of improving both their performance in the serialized setting and the performance of a "truly" integrated system (in which planning and scheduling are more tightly interwoven) which makes use of these solving algorithms.

It should be clear that the N-CBA is certainly not the best way to build an integrated reasoner, since its efficiency relies very strongly on *how separable* the two aspects of the problem are. This requirement is often not realistic, since the degree of inter-dependency between causal and time/resource-related constraints in the problem are much higher. Nonetheless, it should be clear in a moment why we have chosen to start our analysis on this very simple architecture.

The N-CBA has the nice property of delivering a very clear picture of what is going on during problem resolution. In particular, this architecture exposes the following interesting characteristics of planning and scheduling integration:

**Information sharing** The necessity of both a planning and a scheduling solver derives from the fact that they infer different types of information. Thus, it is necessary for the two solvers to exchange *their* view of the problem which is being solved, so as to take advantage of the different perspectives from which the reasoning takes place.

**Deductive bottlenecks** When it comes to solving challenging problems, we are interested in understanding exactly where the difficulties are in the deductive process. In a tightly coupled integration this may be difficult to verify. The analysis of a more loosely coupled architecture like the N-CBA may give interesting indications as to which difficulties can be encountered by the single solving strategies, thus on one hand biasing the choice of the type of strategy, and on the other exposing the necessary adaptation measures to be taken on the strategy.

Let us first of all focus on the information sharing aspect of the N-CBA, which constitutes the effective binding of the planning and scheduling subsystems in any integrated approach.

### 3.1 Information Sharing

The point of integrating planning and scheduling solvers is that they address complementary issues. More specifically, the responsibility of the two solving algorithms is to *validate partial plans with respect to the particular constraints they propagate*. In other words, integrated planning and scheduling systems solve problems by mutually validating the decisions which are taken during partial plan construction with respect to the causal and the time/resource-related constraints.

In general, information sharing between causal and time/resource solvers implements a sort of mutual "advice-giving" mechanism. In this context, the N-CBA is furnished with the most basic form of this type of information sharing, in which the scheduler validates the entire plan, rather than making use of a tight synergy at the partial plan level. Clearly, the ideal form of an information sharing mechanism is one in which this mutual interrogation occurs at every decision point, and, indeed, CSP approaches seem to be moving in that direction.

But even the N-CBA has a rather interesting property, namely that it gives us an indication as to which solving strategies support information sharing mechanisms. To this end, let us see which characteristics the single components of an integrated planning and scheduling system should have.

### 3.2 Solving Strategies

What we have said so far is instrumental in understanding which characteristics we have to look for in order to theorize and implement a more tightly integrated planning and scheduling architecture.

Problems in which scheduling is necessary are typically such that the solution contains concurrent actions (at least at the logical level). In the N-CBA, the scheduling phase is concerned with enforcing the "executability" of a partially ordered plan, so we can be sure that on the causal level there will be a relatively high degree of concurrency to deal with.

The first thing one should notice is that we have taken for granted the use of a PO planner rather than a Total Order (TO) planner as the first component in the N-CBA. This is clearly necessary, given the fact that scheduling is seen as a post-processing phase. Alternatively, the option would have been to cascade a de-ordering phase after the TO planner, in order to establish the concurrency among the actions in the plan for the scheduling phase.The relative convenience of adopting such a strategy is questionable since the computational aspects of de-ordering and re-ordering plans are not trivial. Depending on the criterion with which the de/re-ordering is done, this problem can be NP-Hard [1].

On the other hand, a more realistic approach to planning and scheduling integration is to allow the two solvers to share information in a more constant

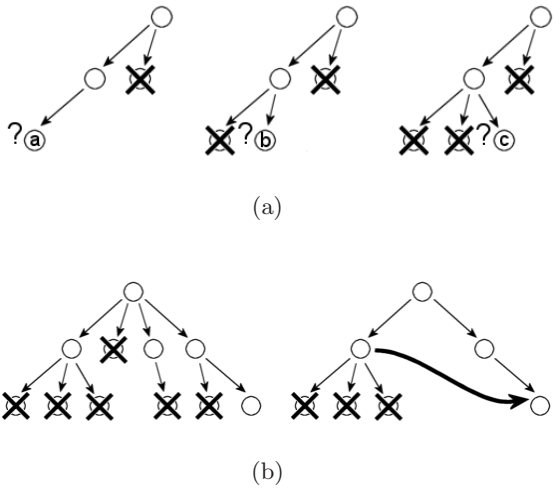



**Fig. 2.** Searching the state space (a) and backtracking vs. de-ordering (b)

fashion during problem solving. In other words, if we see the N-CBA as an approach in which information sharing occurs only in *one* point, namely when the planning phase is done, the most sophisticated form of integration we can think of is one in which such information is exchanged *during* the problem solving. In order to achieve this, the best we can do is to provide the means for each solver to verify *at each decision point* the effects of its decision from the other solver's point of view. Let us see how such an interwoven integration can be realized with a TO solver.

TO planners are typically implemented as Heuristic State Space Search ($HS^3$) algorithms [4, 10, 18]. The state space representation of such planners is very explicit with respect to the decision points of the causal planning phase. Every node represents a "choice" for the planning algorithm, where it relies on a heuristic which gives the solver an indication as to which node to branch over in the following step. For instance, figure 2(a) depicts three successive decisions over the nodes of the state space (nodes represent states while edges represent the actions which lead to them).

In the figure, nodes a, b and c are successively considered with respect to the heuristic and, in an integrated reasoner, also with respect to time and resource feasibility. Now, let us suppose that the subtree which has node c as root is pruned because of time or resource considerations. There are two options for the solver (see figure 2(b)): one is to backtrack in order to explore other regions of the state space, hoping to encounter another node which is time- and resource-feasible. But a more "educated guess" is to de-order the partial plan obtained up to node c. In fact, since the partial plan failed at node c because of time- and resource-related constraints, there is a certain probability that an alternative

ordering of the actions taken so far (one that respects the causal constraints of the domain) could enable the planning to go on. Indeed, this approach is adopted with a certain degree of success in [20].

Notice, though, that we would again have to perform de-ordering. It was to be expected that planning taking into account time and resources is inevitably more expensive computationally that just planning in causal terms. But let us see another way of achieving high levels of integration.

The principal pitfall of the TO-based strategy is that in order for the planner to *see* the time- and resource-feasible node, it must somehow be aware of the existence of the alternative evolution of the state, and this, in turn, requires either de-ordering or backtracking.

So the natural consequence of this is that there is at least one reason for which such a state representation does not seem to match with planning and scheduling integration: the propagation of causal constraints does not take into account all possible evolutions of the state. In other words, the causal decision making during the process of planning is strong to the point that it is retractable only by performing an expensive "roll-back".

It appears obvious that an integrated planning and scheduling architecture must be theorized so as to give *equal importance to both types of deductive processes*. To put it more simply, the reason for which an action can be non-applicable during plan synthesis can be traced back to both causal- *and* time/resource-related considerations. It is clear that the only way of ensuring equal dignity to both solvers is to propagate *sets* of possible choices with respect to causal and time/resource feasibility.

In contrast with $HS^3$ planners, GRAPHPLAN-based planners [3] maintain a planning graph data structure, which represents the *union* of all applicable courses of action given the initial state and the applicable actions. As noted in [12], a planning graph is a sequence of alternated predicate and action levels which correspond to a unioned representation of a state space search tree. The basic solving strategy of GRAPHPLAN-based planners consists of propagating pairs of action and predicate levels in successive steps. At each step, the

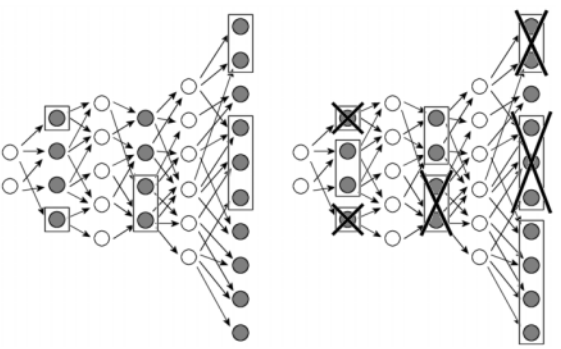


**Fig. 3.** Alternate plans are propagated in the planning graph

planning graph is "searched" for a solution[1]. This representation of the search space is rather different from the tree structure in $HS^3$ planners for one reason in particular: it contemplates *concurrent* actions. In other words, while a tree representation of the state space is fundamentally sequential, a planning graph is capable of representing in a compact way the parallel execution of multiple actions. This means that if some actions which are necessary in order to achieve the goal state do not depend on each other, then they will be present in the solution in the same *logical step*, the solution being a *partial order* of actions.

The planning graph state space representation used in Graphplan-based planning is far more effective from this point of view. A planning graph is a data structure which maintains, as the planning goes along, *all* admissible partial plans (and also non-admissible ones due to the propagation of only pair-wise mutex constraints [12]). Indeed, a planning graph represents the *space of plans* rather than the space of states. Thanks to this, information sharing in a tightly coupled integration based on a Graphplan-like solver is much more realistic. As depicted in figure 3, the exclusion of a partial plan does not require expanding the state space representation, since it already "contains" all alternative partial plans up to the current point of the solving process. Of course, singling out another candidate partial plan is still NP-Hard. But, as we will see in the next section, there is one reason for which this seems to be more advantageous.

### 3.3 BlackBox and HSP

While Graphplan itself can be considered out of its league among the best competition planners, planners such as BlackBox [13] and IPP [14], which are based on the Graphplan paradigm, certainly are not. Our tendency to think that Graphplan-based solvers seem to be better suited for planning and scheduling integration is motivated by the comparison of BlackBox and HSP (a well-known heuristic search planner [4]) on three significant domains taken from the AIPS Planning Competition: Freecell, Rovers and Driverlog.

The experimental base we have considered shows that there is an interesting separation of classical planning domains in two categories: those which can be solved by $HS^3$ planners, and those which are more suited for Graphplan-based planners. Figures 4(a) and 4 show some planning details for problems on the three mentioned domains.

Freecell is a familiar solitaire game found on many computers, involving moving cards from an initial tableau, constrained by tight restrictions, to achieve a final suit-sorted collection of stacks. The problems in this domain show quite a good variety of difficulty, which makes this game rather challenging for most planners.

The first trend we should notice is that BlackBox performs very badly on the Freecell domain. This should not be surprising, since the domain design for this card game is certainly not a parallel one: there is one player, and all actions it

[1] BlackBox extracts solutions by employing SAT algorithms [15, 16] on a CNF formula obtained from the planning graph.

| Problem | BlackBox | HSP |
|---|---|---|
| roverprob1234 | 117 | – |
| roverprob4213 | 102 | – |
| roverprob6232 | 104 | – |
| roverprob2435 | 7882 | – |
| roverprob1423 | 583614 | – |
| roverprob5146 | 17868 | – |
| roverprob1425 | 15482 | – |
| roverprob4135 | – | – |
| roverprob5142 | – | – |
| roverprob5624 | – | – |
| FreeCell2-4 | 119957 | 50 |
| FreeCell3-4 | 1224730 | 230 |
| FreeCell4-4 | – | 470 |
| FreeCell5-4 | – | 1800 |
| FreeCell6-4 | – | 6630 |

(a)

| Problem | BlackBox | HSP |
|---|---|---|
| DLOG-2-2-2 | 29 | 10 |
| DLOG-2-2-3 | 777 | 10 |
| DLOG-2-2-4 | 134 | 20 |
| DLOG-3-2-4 | 1192 | 30 |
| DLOG-3-2-5 | 5805 | 20 |
| DLOG-3-3-5 | 123 | 30 |
| DLOG-3-3-6a | 1550 | 30 |
| DLOG-3-3-7 | 6056 | 50 |
| DLOG-2-3-6a | 8255 | 90 |
| DLOG-2-3-6b | 9427 | 80 |
| DLOG-2-3-6c | 14541 | 200 |
| DLOG-2-3-6d | – | 1890 |
| DLOG-2-3-6e | – | 470 |
| DLOG-3-3-6b | 98677 | 4550 |
| DLOG-4-4-8 | – | 21960 |

(b)

**Fig. 4.** Details for the BlackBox and HSP planning instances for problem instances on the Rovers and Freecell (a) domains, and on the Driverlog (b) domain — time unit is milliseconds

decides to carry out are necessarily sequential (with some, very few exceptions). With some difficulties, BlackBox managed to solve the first two problems, but finally tossed the towel when it came to tackling the more difficult ones. On the other hand, HSP is capable of solving all problems of the Freecell domain rather easily. Notice also that HSP overwhelmingly outperformed BlackBox on all problems.

The Rovers domain models a multi-robot system for exploring an unknown extraterrestrial environment, in which soil-sampling, imaging and data transmission tasks have to be carried out. The problem solving instances described in figure 4(a) show very clearly that this particular domain is rather suitable for planning with BlackBox, which solved most problems rather fast.

For the sake of completeness, we should mention that the same problems are extremely difficult to solve for HSP, which could not cope with any of the them[2] before timing out. Dually with respect to the Freecell problems, we can see that a high degree of concurrency is advantageous for solution extraction, thus making BlackBox quite competitive while planners such as HSP, which do not model resources, are left choking in the dust.

[2] For the first two problems, HSP is not capable of finding a solution and reports this within a few seconds. This is due to the fact that, in this mode of operation, HSP does not ensure completeness.

We have seen sets of problems in two domains, which somehow represent the "extremes" of classical planning: Freecell is a one-player card game, in which practically no parallelism is possible, while the Rovers domain models a highly concurrent application for multiple robots.

This albeit brief experimental evaluation would not be complete without something that resembles a compromise. The well-known Driverlog domain seems to fit the role, namely for its mixed characteristics: on one hand, tasks can be concurrent, such as multiple drivers and trucks involved in the delivery of goods; on the other hand, the domain specifies that trucks have to be loaded and unloaded, and drivers have to reach, board, drive and disembark their trucks. In short, this makes the Driverlog domain not clearly suitable for either of the planning approaches. It turns out, in fact, that most problems are solvable by both BlackBox and HSP, the latter having a slight advantage.

Let us analyze the above domains with respect to their planning and scheduling characteristics. Freecell is a rather "pure" example of a planning domain, given the fact that all actions (moves) are motivated by logical deduction given the situation and the goal or subgoal in a certain logical step. Problems in the Freecell domain do not contemplate any time- or resource-related constraints, nor for that matter is it possible to have two or more actions in parallel since the game is strictly sequential. On the other hand, problems in the Rovers domain are typically scheduling problems in that they could be specified in terms of duration of the actions and their usage of on-board instruments.

A significant trait of the three domains is *parallelism* (vs. *sequentiality*). Clearly, a planning problem specification which allows for multiple tasks to be executed in parallel generally produces *contention peaks* on resources, a problem which is naturally addressed by a scheduler. Thus, such planning problems pose a scheduling problem composed of partially ordered, causally dependent actions which must be instantiated in time so as to satisfy resource capacity constraints.

In the N-CBA it is clear that the preferred planner would be BlackBox, since it performs better than HSP on the problems for which it makes sense to apply an integrated planning and scheduling approach. This result seems to go in the same direction as the considerations we made earlier for tightly integrated systems, namely that in domains which require high concurrency, it is most likely that resource-related considerations are required while planning; because of this, since alternative action orderings are more promptly available thanks to the structure of the planning graph, we can make a case for investigating a Graphplan-based approach to planning and scheduling integration.

## 4 The N-CBA as a Lower Bound

So far we have given a glimpse of the issues which have to be taken into consideration in the theorization of an integrated planning and scheduling solution. From the previous considerations we can derive that not all planning paradigms are well suited for an integrated approach.

An integrated planning and scheduling reasoner built according to the N-CBA is reported in [19]. In this section we will give an overview of this implementation of the N-CBA which makes use of established off-the-shelf components for the planning and scheduling modules. The choice of these components (or better, of the planning system) is strongly grounded on the previous considerations. Our aim is to assert a basic loosely-coupled system according to the N-CBA, which can be to some extent taken as the "best we can do" with the most primitive form of information sharing.

The planning system which has been used is BlackBox, a Graphplan-based planner which combines the efficiency of planning graphs with the power of SAT solution extraction algorithms [15, 16]. This planner has done very well in recent planning competitions, and in the category of those planning paradigms which seem fit for planning and scheduling integration, it can certainly be considered to be one of the best choices. The scheduler which has been selected for use in this implementation of the N-CBA is O-Oscar [7], a versatile, general purpose CSP solver which implements the ISES algorithm [6].

The result is an architecture which is effectively capable of solving quite a number of problems, and which has been extensively used in the development of a preliminary Active Supervision Framework [19] for the RoboCare project. It is not trivial to notice that even this primitive form of integration has some important advantages:

- the only necessary additional development is the adaptation procedure, in other words, the information sharing mechanism;
- assuming the separability of causal and time/resource-related problem specifications, the system performs quite well, especially with respect to plan quality;
- it is a general purpose tool, thanks to the PDDL planner input, and to the high expressivity of the scheduling problem specification [5, 2];

The performance details of this implementation of the N-CBA are outside the scope of this paper. These issues are extensively reported in [19], in which the N-CBA has been used to solve problems on a multi-agent domain for a health care institution.

Clearly, a more tightly integrated system is often necessary, especially because causal and time/resource-related aspects of the domain are often strongly dependent. Nonetheless, it is interesting to notice how the basic properties of solving ideas and their relative adherence to the philosophy of integration are somehow independent from *when* the information sharing occurs (at partial plan level, at every decision point and so on), rather they depend on *which* information is shared between the two solving engines. It is clear already in the N-CBA that the most effective solution seems to be that of adopting strategies which are capable of propagating plans rather than states, in order to ensure fast reactivity when it comes to taking decisions.

In the previous sections we have motivated the use of a Graphplan-based planner in conjunction with a CSP scheduling tool. The most primitive approach

to planning and scheduling integration is clearly the N-CBA, an approach in which information sharing occurs only once during the solving process. Given the coarseness of this integrated solving strategy, our opinion is that more sophisticated implementations of integrated solvers should be compared to the N-CBA[3]. In fact, the overall performance of an integrated architecture can only get better with higher occurrences of information sharing events between the two solvers.

## 5 Conclusions and Future Work

In this paper we have presented an analysis of the issues involved in planning and scheduling integration. Many approaches to this problem have been developed, yet some fundamental aspects still remain to be explored fully. The work we have presented is motivated by the fact that while the independent research in planning and scheduling has evolved very rapidly, the study of the combined approach is still trying to deal with some basic issues.

The intent of this paper is to analyze some of these issues starting from the N-CBA, an approach to integration which implements the simplest form of information sharing. Integrated architectures in general can be distinguished by two parameters: *when* information sharing is carried out, and *what* information is shared between the two solvers. We have shown how higher degrees of integration are obtainable from the N-CBA by "toggling" the first parameter, where the optimal strategy is to cross-validate causal and time/resource-related aspects of the problem at every decision point.

The study of the N-CBA is interesting because it exposes some characteristics which are common to all forms of integration. In particular, we have focused on which solving strategies are more applicable in an integrated solver. In this context, we have found that both in the N-CBA and in more tight integrated approaches, a desired property of the solvers is to propagate sets of possible choices with respect to causal and time/resource feasibility.

The analysis of the N-CBA has shown how a variety of useful indications can be drawn with respect to the general issue of planning and scheduling integration: thanks to these considerations, we are aiming at furthering the study in this direction, looking in particular at CSP approaches to planning.

## Acknowledgements

This research is partially supported by MIUR (Italian Ministry of Education, University and Research) under project RoboCare (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components). The Authors are part of the Planning and Scheduling Team [PST] at ISTC-CNR and would like to thank the other members of the team for their continuous support.

[3] Assuming the two aspects of the problem are distinguishable in the benchmark domains.

## References


[1] Bäckström, C. Computational Aspects of Reordering Plans. *Journal of Artificial Intelligence Research 9* (1998), 99–137. 364, 365

[2] Bartusch, M., Mohring, R. H., and Radermacher, F. J. Scheduling Project Networks with Resource Constraints and Time Windows. *Annals of Operations Research 16* (1988), 201–240. 371

[3] Blum, A., and Furst, M. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* (1997), 281–300. 367

[4] Bonet, B., and Geffner, H. Planning as heuristic search. *Artificial Intelligence 129*, 1–2 (2001), 5–33. 366, 368

[5] Brucker, P., Drexl, A., Mohring, R., Neumann, K., and Pesch, E. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operations Research* (1998). 371

[6] Cesta, A., Oddi, A., and Smith, S. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics 8*, 1 (2002), 109–135. 371

[7] Cesta, A., Oddi, A., and Susi, A. O-Oscar: A Flexible Object-Oriented Architecture for Schedule Management in Space Applications. In *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-99)* (1999). 371

[8] Currie, K., and Tate, A. O-Plan: The Open Planning Architecture. *Artificial Intelligence 52*, 1 (1991), 49–86. 363

[9] Ghallab, M., and Laruelle, H. Representation and Control in IxTeT, a Temporal Planner. In *Proceedings of the Second International Conference on AI Planning Systems (AIPS-94)* (1994). 363

[10] Hoffmann, J., and Nebel, B. The FF Planning System: Fast Plan Generation Through Heuristic Search, 2001. 366

[11] Jonsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)* (2000). 363

[12] Kambhampati, S., Parker, E., and Lambrecht, E. Understanding and Extending Graphplan. In *Proceedings of ECP '97* (1997), pp. 260–272. 367, 368

[13] Kautz, H., and Selman, B. Unifying SAT-Based and Graph-Based Planning. In *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999* (College Park, Maryland, 1999), J. Minker, Ed., Computer Science Department, University of Maryland. 368

[14] Koehler, J., Nebel, B., Hoffmann, J., and Dimopoulos, Y. Extending Planning Graphs to an ADL Subset. In *Proceedings of ECP '97* (1997), pp. 273–285. 368

[15] McAllester, D., Selman, B., and Kautz, H. Evidence for Invariants in Local Search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)* (Providence, Rhode Island, 1997), pp. 321–326. 368, 371

[16] Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., and Malik, S. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)* (2001). 368, 371

[17] Muscettola, N., Smith, S., Cesta, A., and D'Aloisi, D. Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture. In *IEEE Control Systems, Vol.12, N.1* (1992), pp. 28–37. 363

[18] Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Longman Publishing Co., Inc., 1984. 366

[19] Pecora, F., and Cesta, A. Planning and Scheduling Ingredients for a Multi-Agent System. In *Proceedings of UK PLANSIG02 Workshop, Delft, The Netherlands* (2002). 371

[20] R-Moreno, M., Oddi, A., Borrajo, D., Cesta, A., and Meziat, D. Integrating Hybrid Reasoners for Planning & Scheduling. In *Proceedings of UK PLANSIG02 Workshop, Delft, The Netherlands* (2002). 367

# Evolving the Neural Controller for a Robotic Arm Able to Grasp Objects on the Basis of Tactile Sensors

Raffaele Bianco and Stefano Nolfi

Institute of Cognitive Science and Technologies, National Research Council (CNR)
Viale Marx, 15, 00137, Roma, Italy
{rbianco,nolfi}@ip.rm.cnr.it

**Abstract.** We describe the results of a set of evolutionary experiments in which a simulated robotic arm provided with a two-fingers hand has to reach and grasp objects with different shapes and orientations on the basis of simple tactile information. Obtained results are rather encouraging and demonstrate that the problem of grasping objects with characteristics that varies within a certain range can be solved by producing rather simple behavior that exploit emergent characteristics of the interaction between the body of the robot, its control system, and the environment. In particular we will show that evolved individuals does not try to keep the environment stable but on the contrary push and pull the objects thus producing a dynamics in the environment and exploit the interaction between the body of the robot and the dynamical environment to master rather different environmental conditions with rather similar control strategies.

## 1 Introduction

The problem of controlling a robotic arm is usually approached by assuming that the robot should have or should acquire through learning an internal model able to: (a) predict how the arm will move and the sensations that will arise, given a specific motor command (direct mapping), and (b) transform a desired sensory consequence into the motor command that would achieve it (inverse mapping), for a review see [1]. We do not deny that humans and other natural species rely on internal models of this form to control their motor behavior. However, we do not believe that motor control and arm movement in particular are based on a detailed description of the sensory-motor effects of any given motor command and of a detailed specification of the desired sensory states.

Assuming that natural organisms act on the basis of a detailed direct and inverse mapping is implausible for at least two reasons. The first reasons is that sensors only provide incomplete and noisy information about the external environment and muscles have uncertain effects. The former aspect makes the task of producing a detailed direct mapping impossible given that this would require a detailed description

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 375-384, 2003.
© Springer-Verlag Berlin Heidelberg 2003

of the actual state of the environment. The latter aspects makes the task of producing an accurate inverse mapping impossible given that the sensory-motors effects of actions cannot be fully predicted. The second reason is that the environment might have its own dynamic and typically this dynamic can be predicted only to a certain extent. For these reasons, the role of the internal models is probably limited to the specification of macro-actions or simple behaviors rather than to micro-actions that indicate the state of the actuators and the predicted sensory state in any given instant.

This leave the question of how macro-actions or simple behaviors might be turned into micro-actions open. One possible solution to this problem is to imagine that the ability to produce macro-actions (i.e. basic motor behaviors such as grasping a certain class of objects in a certain class of environmental conditions) are produced through simple control mechanisms that exploit the emergent result of fine grained interactions between the control system of the organism, its body and the environment.

To investigate this issue we run a set of experiments in which we evolved the control system of a simulated robotic arm provided with a two-fingers hand that has to reach and grasp objects with different shapes and orientations on the basis of simple tactile information. As we will see, evolving individuals develop an ability to grasp objects in different environmental conditions without relying on direct and inverse mappings.

In section 2 we describe the related work, in section 3 and 4 we present our experimental setup and the results obtained by evolving robotic arm in different experimental conditions. In section 5 we discuss the obtained results and their implications.

## 2 Related Work

As far as we know, there have been only two attempts to apply evolutionary robotics techniques [2] to the synthesis of robotic arms. The first attempt has been done by Moriarty and Mikkulainen [3] who evolved the neural controller of a robotic arm with three degrees of freedom in simulation. The arm is initially placed in a random position and is asked to reach a random target position by avoiding obstacles. At any time step the neural controller receives as input the relative distance between the hand and the target position with respect to the three geometrical axis (x,y,z) and the state of 6 directional proximity sensors located in the hand. The robotic arm and the environment are simulated in a rather simplified way (e.g. collisions between objects are not simulated -- the authors simply stop the arm when its end point moves into a position occupied by an obstacle). The second attempt has been done by Skopelitis [4] who evolved the control system of a robotic arm with three degrees of freedom that is asked to follow a moving target. At any time step the neural controller receives as input the (x,y,z) coordinate of the target, the coordinate of the hand, the Euclidean distance of the hand and the target, and the coordinates of the "elbow" joint. Also in this case, the robotic arm and the environment are simulated in a rather simplified way (e.g. the target is not a physical object but only an abstract point of the environment and the arm has no mass and is not subjected to physical forces or collisions).

In the experiments reported in this article we tried to evolve the neural controller for a much more complex robotic arm provided with a two fingers hand that is asked to grasp objects. The arm and the environment are carefully simulated (see below) and the controller is only provided with simple touch and proprioceptive sensors (i.e. it does not have access to information that cannot be computed by local sensors such us the distance with respect to the target position).

# 3 The Robotic Arm and the Neural Controller

## 3.1 The Robotic Arm

The robot consists of an arm with six degree of freedom (DOF) and a two fingers hand provided with three DOF (Figure 1, left). The arm consists of three connected basic structures forming two segments and a wrist. Each basic structure consists of two bodies connected by two motorized (Figure 1, centre). More precisely, each basic structure (Figure 1, bottom-right) consists of a parallelepiped with a size of [x=50, y=30, z=50] cm and a weight of 750 g and a cylindrical object with a radius of 25 cm, a length of 1 m, and a weights of 1.96 kg (a length of 20 cm and a weight of 390 g in the case of the last cylindrical object that forms the base of the hand). Parallelepipeds are connected to the previous segment or to a fixed point (in the case of the first segment) by means of a rotational joint (R_Joint) that provided one DOF on axis Y. Cylinders are connected with parallelepipeds by means of an elevation joint (E_Joint) that allows only one DOF on axis Z. In practice, the E_Joint allows to elevate and lower the next connected segments and the R_Joint allows to rotate them in both direction. Notice that E_Joint is free to moves only in a range between 0 and $\pi/2$, just like an human arm that can bend the elbow solely in a direction. The range of R_Joint is $[-\pi/2, +\pi/2]$ for the first two and is $[0, \pi]$ for the last basic structures.

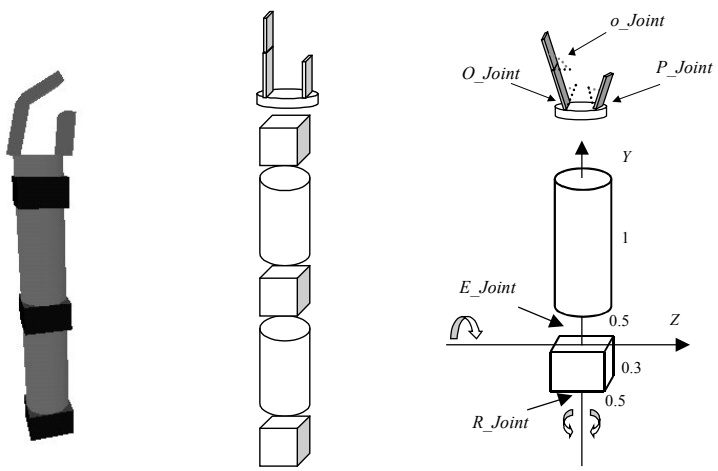



**Fig. 1. Left**: The arm and the hand. **Centre**: A schematic description of the elements forming the arm and the hand. **Right**: A schematic description of the motorized joints that connect the different element of the arm and of the hand

The hand consists of two fingers made of two parallelepipeds with a size of [x=10, y=40, z=40] cm and a weight of 880 g connected by two motorized joints (*O_Joint* and *P_Joint*) to the last cylindrical object forming the arm. These two joints, that allow to open and close the two fingers, can move only in a range of [-π/10, π/6] and [-π/10, π/4] respectively (Figure 1, right up). The first finger has an additional phalange consisting of a parallelepiped with a size of [x=10, y=40, z=50] cm and a weight of 1.1 Kg connected by a motorized joints (*o_Joint*) to the previous part of the finger. This additional joint that allows the finger to close its upper part can move in the range [0, π/2].

Each actuator is provided with a corresponding motor that can apply a maximum force of 0.3 dynes-centimetre for the segments joints and 0.15 dynes-centimetre for the fingers joints. Friction coefficient is set to 0.7 and the acceleration of gravity is – 0.098 $m/ds^2$.

This means that to reach and grasp an object the robot has to appropriately control 9 joints and to deal with the constraints due to gravity and collisions.

The sensory system consists of six contact sensors (three placed on the three cylindrical objects forming the arm and the wrist and three placed on the three parallelepipeds forming the two fingers) that detect, in a binary fashion, whether these bodies collide with other bodies. Moreover, robots have nine proprioceptive sensors that encode the current angular position of the nine corresponding motor joints controlling the arm and the fingers.

The environment consists of a planar surface (at height 0) and an object (e.g. a ball, a cube, or a bar) placed on the surface (see Figure 3). The first element of the arm is anchored to a fixed point [x=0, y=230, z=0] cm and is oriented along the vector [0,-0.86,-0.5].

To reduce the time necessary to test individual behaviors and to model the real physical dynamics as accurately as possible we used the rigid body dynamics simulation SDK of Vortex™ (see http://www.cm-labs.com/products/vortex/). This software allowed us to a build a rather realistic simulation and to speed up the evolutionary process by allowing simulated robots to move faster than real physical robots.

### 3.2 The Neural Controller

Each individual is controlled by a fully connected neural network with 15 sensory and 9 motor neurons. Neurons are updated with the logistic function.

The sensory neurons encode the angular position (normalized between 0.0 and 1.0) of the 9 DOF of the joints and the state of the six contact sensors located in the arm and in the fingers. The motor neurons control the actuators of the 9 corresponding joints. The output of the neurons is normalized between the range of movement of the corresponding joint and is used to encode the desired position of the corresponding joint. Motors are activated so to reach a speed proportional to the difference between the current and the desired position of the joint.

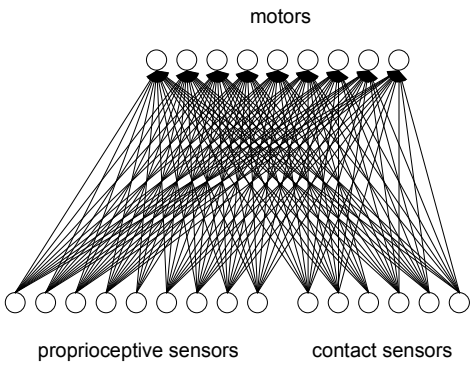



**Fig. 2.** The architecture of the neural controller

The genotype of evolving individuals encode the connections weights and the biases of the neural controller. Each parameter is encoded with 8 bits. Weights and biases are normalized between –10.0 and 10.0. Population size was 100. The 20 best individuals of each generation were allowed to reproduce by generating 5 copies of their genotype which were mutated by replacing 2% of randomly selected bits with a new randomly chosen value. Each experiment was replicated 10 times.

Each individual of the population was tested for a given number of trials, with each trial consisting of a given number of steps (each steps lasts 200 ms of real time). At the beginning of each trial the starting angle of the three E_Joints are set to 0, 70 and 30 degrees respectively, for R_Joints are set to 0, -30 and 20 degrees, for O_Joints, o_Joint and P_Joint is set to 0 degrees (see Figure 3).

To measure the ability of evolving individuals to grasp objects we removed the plane supporting the objects after a fixed number of steps in every trial and rewarded individuals on the basis of the number of objects that did not fall down at the end of each trial. In addition, to facilitate the emerge of an ability to grasp objects, we also rewarded individuals for their ability to touch the object with their fingers (i.e. we use a form of incremental evolution, see Nolfi and Floreano [2000]). More precisely the fitness of an individual was computed according to the following equation:

$$\text{fitness} = (GP * 10000) + NC$$

where *GP* is the number of objects that have been successfully grasped (i.e. that have an y coordinate higher or equal to 50 cm and that collided with at least one of the finger at the end of the corresponding trial), and *NC* is the number of collision between the objects and the robotic fingers during the whole lifetime of an individual.

# 4 Experimental Results

## 4.1 Grasping Cubic or Spherical Objects with Different Size, Weight and Orientation

In a first set of experiments we asked evolving individuals to grasp cubic or spherical objects with different size, weight and orientation. Each individuals was tested for 20

trials, with each trial consisting of 200 steps (the plane is removed after 150 steps). During its life each individual experienced 10 spherical and 10 cubic objects that were located in the following position [x=200, z=0] cm (the y coordinate was set proportionally to the size so to assure that the object lay on the plane). The size, weight, and orientation of the objects was randomly chosen in each trial within a given range. The side of cubic objects varied between 30 and 40 cm and the radius of spherical objects varied between 20 and 30 cm. The density of the objects vary between 10 and 50 kg/m$^3$. The orientation varied between 0 and 90 degrees along the Y axis.

By running 10 evolutionary experiments for 50 generations we observed (see Figure 3) that evolving individuals display rather good performance (up to 100% of successful trials in the case of the best replication and up to 90% of successful trials on the average).

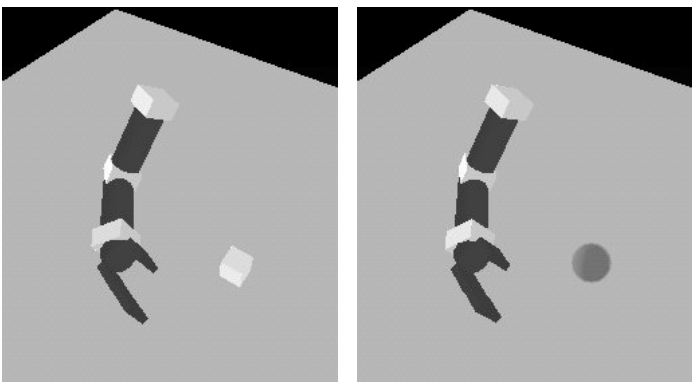


**Fig. 3.** The arm, in its initial position, and the environment. The environment consists of a cubic or a spherical object laying on the plane

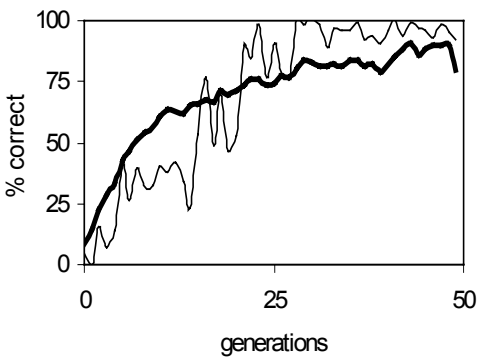



**Fig. 4.** Percentage of objects correctly grasped obtained by testing the best individuals of each generation for 100 trials. **Thin line**: performance in the case of the best replication. **Thick line**: average performance of 10 replications

The analysis of the behaviour displayed by evolved individuals shows that evolved robots are able to reach and grasp different type of objects with different sizes and weights by mastering the dynamical interaction between the objects and the hand.

Indeed, as shown in Figure 5 that displays the behaviour of a typical evolved individual, the robot approaches the object with the hand opened from the top-left side of the object and then starts to close the hand as soon as it detects the object with the contact sensors of the fingers. However, while grasping the object, the robot also rotates the hand on the right side. This movement from left to right allows the robot to block the movement of the object (caused by the collision between the hand and the object) with the hand itself and allows the robot to exploit the properties emerging from the dynamical interaction between the moving object and the hand. Indeed, as a result of the initial collision between the hand and the object and of the successive rotation of the hand, objects tend to move toward the inner part of the hand so to spontaneously adjust small misplacements resulting from the fact that objects have different sizes and orientations.

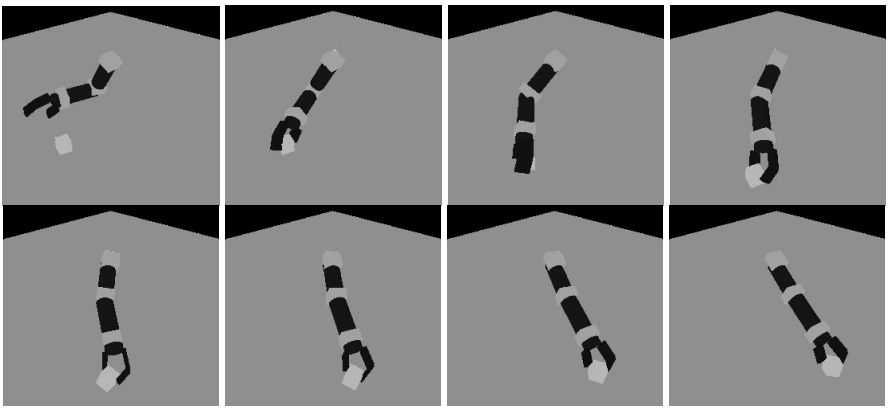


**Fig. 5.** A typical behaviour displayed by an evolved individual. The eight pictures (from left to right and from top to bottom) show eight snapshots of a trial in which the robot approach and grasp a cubic object

### 4.2 Grasping Bars with Different Weight and Orientation

In a second set of experiments we asked evolving individuals to grasp bars with different weight and orientation (Figure 6). Given that bars can only be grasped by placing the hand in the right relative orientation with respect to the object, we might expect that to solve this problem robots should first detect the orientation of the bar and then approach the object appropriately. As in the case of the previous experiment, however, by exploiting the interaction between the hand and the object evolving individual develop a simpler solution that consist in modifying the orientation of the bar.

Each individuals was tested for 30 trials, with each trial consisting of 250 steps (the plane is removed after 200 steps). In each trial the barycentre of the bar is initially placed in the position [x=200, y=15, z=0] but the orientation is randomly chosen between 0 and 180 degrees along the Y axis and the weight randomly varies between 10 and 20 kg/m$^3$ (see Figure 6). The evolutionary process was continued for 100 generations. All other parameters are identical to that of the experiment described in section 2.1.

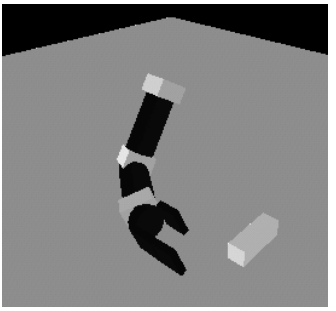


**Fig. 6.** The arm and the bar at the beginning of a trial. The orientation and the weight of the bar is randomly chosen in every trial within a given range (see text)

Also in this experiment evolving individuals display rather good performance (up to 94% of successful trials in the case of the best replication and up to 77% of successful trials on the average) (see Figure 7).

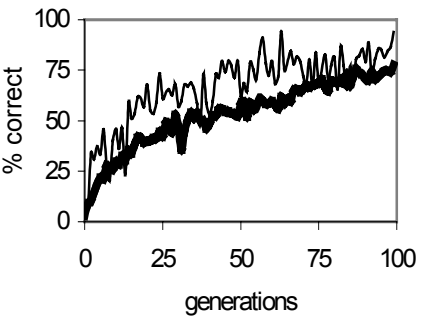



**Fig. 7.** Percentage of objects correctly grasped obtained by testing the best individuals of each generation for 100 trials. **Thin line**: performance in the case of the best replication. **Thick line**: average performance of 10 replications

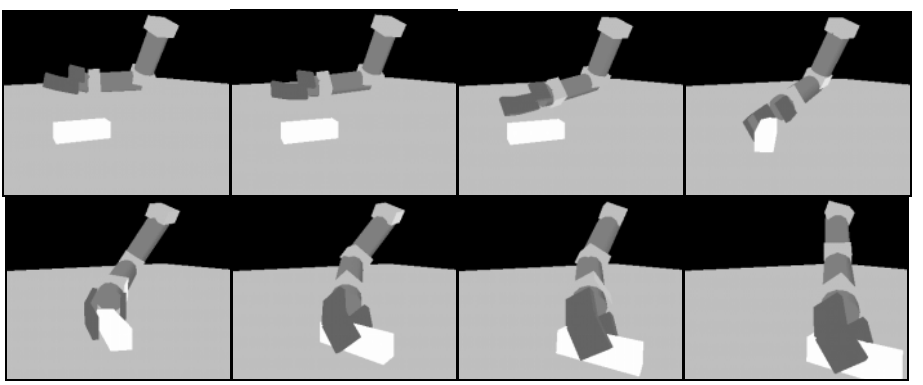


**Fig. 8.** A typical behaviour displayed by an evolved individual. The eight pictures (from left to right and from top to bottom) show eight snapshots of a trial in which the robot approach and grasp a bar with a randomly selected orientation

The analysis of the behaviour displayed by evolved individuals shows that evolved robots are able to reach and grasp bars independently from their relative orientation and weight most of the times. As shown in Figure 8 that displays the behaviour of a typical evolved individual, the robot approaches the object from the left side and, while grasping the object, also rotates the bar toward the preferred orientation (in the case of this individual, the orientation that the bar has in the bottom-right picture). In other words, robots do not need to detect the current orientation of the bar and then approach the object from different orientations according to the actual position of the bar and can solve the problem with a rather simple behaviour by exploiting the dynamical interaction between the hand and the environment. In particular, the bar-rotation behaviour emerges from the simple approaching behaviour produced by the robot, the different length of the two fingers, and the effect of the collisions between the hand and the bar produced by the movements of the arm and of the fingers.

## 5 Discussion

In this paper we present a set of evolutionary robotics experiments in which simulated robotic arms provided with a two-fingers hand develop an ability to reach and grasp objects with different locations, shapes and orientations on the basis of simple tactile information. Obtained results are rather encouraging and demonstrate that the problem of grasping objects with characteristics that varies within a certain range can be solved by producing rather simple behavior that exploit emergent characteristics of the interaction between the body of the robot, its control system, and the environment. In particular we showed that in all cases, evolved individuals does not try to keep the environment stable but on the contrary push and pull the objects thus producing a dynamics in the environment and exploit the interaction between the body of the robot and the dynamical environment to master rather different environmental conditions with rather similar control strategies.

The results of these experiments demonstrate that evolutionary robotics is an ideal framework for synthesizing embodied and situated agents in which behavior is an emergent result of the dynamical interaction between the nervous system, the body, and the external environment (see also [5-6]). Moreover, they demonstrate that the evolutionary robotic approach can scale up to the synthesis of robots with many degrees of freedom that are able to operate robustly in different and dynamical environmental conditions. Indeed, the ability to exploit properties that emerge from the dynamical interaction between the control system of the robot, its body, and the external environment allows the evolutionary process to find solutions that are very simple from the point of view of the control system.

Finally, by trying to evolve robots able to grasp objects in different conditions we observed that some problems such us the problems of grasping objects with different weight, shape, and orientation could be solved by relying on extremely simple neural controllers in which sensory neurons were directly connected to motor neurons. In these cases, as we claimed above, evolving individuals exploit the dynamical interaction between the body of the robot and the environment in order to master rather different environmental conditions with rather simple control strategies.

## References

[1] Torras C. (2002). Robot arm control. In M.A. Arbib (Ed.) *The Handbook of Brain Theory and Neural Networks*, Second edition. Cambridge, MA: The MIT Press.

[2] Nolfi S. & Floreano D. (2000). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Cambridge, MA: MIT Press/Bradford Books

[3] Moriarty D.E. and Mikkulainen R. (1996). Evolving obstacle avoidance behavior in a robot arm. In Maes P., Mataric M., Meyer J.-A., Pollack J., and Wilson S.W (eds.), *Proceedings of the Fourth International Conference on Simulation of Adaptive Behaviors*. Cambridge MA: MIT Press.

[4] Skopelitis C. (2002). Control System for a Robotic Arm. *Master Thesis*. School of Cognitive and Computing Sciences (COGS), University of Sussex, U.K.

[5] Nolfi S. & Marocco D. (2001). Evolving robots able to integrate sensory-motor information over time. *Theory in Biosciences,* 120:287-310.

[6] Nolfi S. (2002). Power and Limits of Reactive Agents. *Neurocomputing*, 42:119-145.

# An Early Cognitive Approach to Visual Motion Analysis

Silvio P. Sabatini and Fabio Solari

Department of Biophysical and Electronic Engineering
University of Genova, Via Opera Pia 11/a, 16145 Genova, Italy
{silvio,fabio}@dibe.unige.it
http://www.pspc.dibe.unige.it

**Abstract.** Early cognitive vision can be related to the segment of perceptual vision that takes care of reducing the uncertainty on visual measures through a visual context analysis, by capturing regularities over large, overlapping retinal locations, a step that precedes the true understanding of the scene. In this perspective, we defined a general framework to specify context sensitive motion filters based on elementary descriptive components of optic flow fields. The resulting regularized patch-based motion estimation obtained in real-world sequences validated the approach.

## 1 Introduction

Computer vision procedes through several stages, ranging from low-level (early vision) processes, mainly devoted to feature extraction, to high-level (visual cognitive) processes, dedicated to recognition and dynamic 3D shape inference, up to the extraction of spatio-temporal relationships between the perceptual agent and the scene's objects. In general, there is a gap between early and cognitive vision paradigms. This gap is not only due to their different position in the hierarchical bottom-up scheme of visual processing, but also relates to the different computational paradigms they adopt. Early vision processes are usually based on distributed computation (cf. parallel distributed processing), that can be directly associated to neuronal mechanisms (cf. neuromorphic approach). On the other hand, cognitive processes are traditionally associated to the AI approach, based on symbolic processing and logic, operating in terms of symbols and propositions, and aimed to the understanding of the scene. This leads to systems in which visual feature (like edges, depth, motion, etc.) are computed from early-vision algorithms and those features are then subjected to a relational analysis. In this way, there is a risk of "jumping to conclusions", leaving a distributed representation of visual features too fast, for an hazardous integrated description of cognitive entities. Considering that visual features computed from early vision algorithms are usually error-ridden, it is rather complicated to subject them directly to a relational analysis. Each measure of an observable property of the visual stimulus is, indeed, affected by an uncertainty (not only due to the additive noise, but also to the fact that the visual properties are themselves

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 385–397, 2003.
© Springer-Verlag Berlin Heidelberg 2003

random processes) that can be removed, or, better, reduced by making use of additional information (context information, a priori knowledge, etc.). *Early cognitive vision* can be related to the segment of perceptual vision that takes care of reducing the uncertainty on visual measures through a visual context analysis, that is by capturing coherent properties (regularities) over large, overlapping retinal locations (Gestalts [1]), a step that precedes the true understanding of the scene. Following a conventional AI approach, the use of contextual information occurs through the application of specific knowledge-based rules to establish consistency relationships among the extracted visual features. By contrast, in visual cortex, contextual modulation of the sensorial input occurs through dense intra- and inter-area feedback interconnections that integrate context information by modulating cells' responses, adapting their tuning and refining their selectivity. We challenged the goal of mimicking cortical computational paradigms to develop parallel distributed processing systems to implement adaptive visual filters, which are fully data-driven and avoid explicit use of AI rules. This would allow to define *context-sensitive filters* (CSFs) based on structural computation rather than on mere calculus. It is important to stress the necessity of re-thinking about cognitive aspects in *structural* terms, by evidencing novel strategies to allow a more direct (i.e., structural) interaction between early vision and cognitive processes, that can be employed by new artificial vision systems. In this perspective, we defined a general framework to specify context sensitive motion filters based on deterministic (i.e., geometric) spatial motion Gestalts. In particular, the geometric properties of the optic flow field have been described through a specific set of elementary gradient-type patterns, as cardinal components of a linear deformation space. By checking the presence of such Gestalts in optic flow fields, we make the interpretation of visual motion more confident. Given motion information represented by an optic flow field, we recognize if a group of velocity vectors belong to a specific pattern, on the basis of their relationships in a spatial neighborhood. Casting the problem as a Kalman filer (KF), the detection occurs through a spatial recurrent filter that checks the consistency between the spatial structural properties of the input flow field pattern and a structural rule expressed by the process equation of the Kalman filter.

## 2 A Kalman Filtering Approach to Early-Cognitive Vision

**Basic Concepts** Perception can be viewed as an inference process to gather properties of real-world, or *distal*, stimuli (e.g., an object in space) given the observations of *proximal* stimuli (e.g., the object's retinal image). In this perspective, early cognitive vision can be cast as an *adaptive filter* in which some kind of early-cognitive algorithm plays the role of the *adaptive process*. A general adaptive filtering system is shown in Fig. 1, where $\boldsymbol{x}^*[k]$ is the *unknown*

[1] In this paper, Gestalts are defined as pixel groups with a shared and persistent properties in space and/or time. This concept goes beyond that of a visual "feature", because Gestalts capture the relationships existing among features.

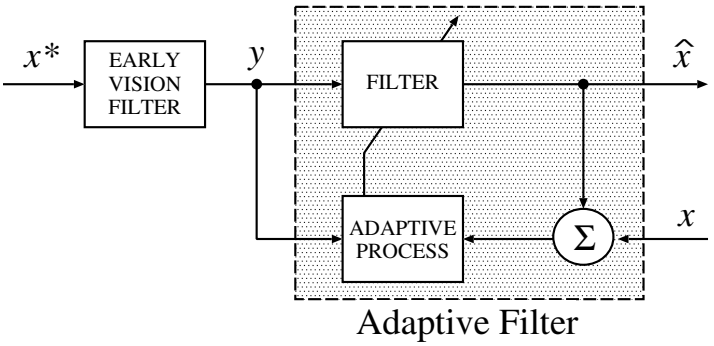



**Fig. 1.** Schematic representation of an adaptive early vision filter

stimulus (the *state*) at time step $k$, $\boldsymbol{y}[k]$ is the observation of the stimulus (the *measure*), $\hat{\boldsymbol{x}}[k]$ is the estimated stimulus, and $\boldsymbol{x}[k]$ is the reference signal (i.e., what we know about $\boldsymbol{x}^*[k]$). The purpose of a general adaptive system is to filter the input signal $\boldsymbol{y}[k]$ (measure) to invert (in some sense) the measure operator and gain an estimation of $\boldsymbol{x}^*[k]$ by making use of the knowledge $\boldsymbol{x}[k]$. Such a knowledge can be provided by:

1. the *visual context*
   - the relationships among the feature values of a single modality in a spatial neighborhood (e.g., responses outside the early vision filter): spatial context;
   - the relationships among the feature values of a single modality in a (spatio-)temporal neighborhood (e.g., the constraints posed by rigid body motion): (spatio-)temporal context;
   - the interdependences of punctual/local feature values from different modalities: multimodal context;
2. the *state of the perceptual agent* (e.g., alert state, task dependency, expectation, etc.)
3. *a priori* information (e.g., cognitive models such as shading, familiarity, perspective, etc.)

In this paper we focus only on data-driven (exogenous) information provided by the visual context, disregarding the other two model-driven (endogenous) components.

**Kalman Estimator** The Kalman Filter is an optimal recursive linear estimator [1], in the sense that it can iteratively process new measures as they arrive, on the basis of the knowledge about the system accrued by previous measurements. Accordingly, a recursive process equation is required to describe the reference signal (the model). Due to its recurrent formalization it appears particularly promising to design *context-sensitive filters* based on recurrent cortical-like interconnection architectures. Formally, the two inputs to the filter are:

the *process equation*

$$\boldsymbol{x}[k] = \boldsymbol{\Phi}[k, k-1]\ \boldsymbol{x}[k-1] + \boldsymbol{S}[k-1]\ \boldsymbol{s}[k-1] + \boldsymbol{n}_1[k-1] \tag{1}$$

and the *measurement equation*

$$\boldsymbol{y}[k] = \boldsymbol{C}[k]\ \boldsymbol{x}[k] + \boldsymbol{n}_2[k] \tag{2}$$

The matrix $\boldsymbol{\Phi}[k, k-1]$ is a known state transition matrix that relates the state at the previous time step $k-1$ to the state at the current step $k$. The matrix $\boldsymbol{S}[k]$ takes into the account an optional control input to the state. The matrix $\boldsymbol{C}[k]$ is a known measurement matrix. The process and measurement uncertainty are represented by $\boldsymbol{n}_1[k] = N(0, \boldsymbol{\Lambda}_1[k])$ and $\boldsymbol{n}_2[k] = N(0, \boldsymbol{\Lambda}_2[k])$. The space spanned by the observations $\boldsymbol{y}[1], \boldsymbol{y}[2], \cdots, \boldsymbol{y}[k-1]$ is denoted by $\boldsymbol{\mathcal{Y}}_{k-1}$.

**Casting** Let us interpret the meaning of the input/output signals of the KF in relation with our perceptual problem.

*Measurement equation* - The linear operator $\boldsymbol{C}$ represents a general "early-vision filter" providing a noisy measure of an observable property of the visual stimulus.

*Process equation* - Assuming $\boldsymbol{x}$ a vector containing the values of a bunch of visual features over a fixed spatial region, Eq. 1 models the temporal evolution of the relationships among such features, according to specific rules embedded in the transition matrix $\boldsymbol{\Phi}$. By example, if we consider just one feature (e.g., motion velocity), $\boldsymbol{x}[k]$ will represent the "model" optic flow values at time step $k$, for all the (discrete) locations of the considered spatial regions (the velocity state). If $\boldsymbol{\Phi}$ has a diagonal structure, the process equation will describe the "model" temporal evolution of punctual velocities, independently of the spatial neighborhood values (temporal context). On the other hand, if $\boldsymbol{\Phi}$ shows a non-diagonal structure, the process equation models a "model" temporal evolution of the state that takes into account *also* spatial relationships (spatio-temporal context). More generally, if we build a state vector that collects more multiple features (e.g., motion, stereo, etc.), by proper specification of the transition matrix $\boldsymbol{\Phi}$, the process equation can potentially model any type of *multimodal* spatio-temporal relationships (multimodal context).

*Filter output* - Apart from the KF output $\hat{\boldsymbol{x}}$, we could be interested in making the measurements more confident. Accordingly, the output will be $\hat{\boldsymbol{y}}[k|\boldsymbol{\mathcal{Y}}_k]$, to be compared with $\boldsymbol{y}[k]$. The additional (contextual) information will be provided by Kalman innovation. We expect that, if the model is correct, the uncertainty associated to the *a posteriori* estimate of the actual measure $\hat{\boldsymbol{y}}[k|\boldsymbol{\mathcal{Y}}_k]$ is inferior to the uncertainty associated to the actual measure itself $\boldsymbol{y}[k]$.

## 3 Motion Gestalts

Local spatial features around a given location of a flow field, can be of two types: (1) the average flow velocity at that location, and (2) the structure of the local variation in a the neighborhood of that locality [2]. The former relates to

the *smoothness constraint* or *structural uniformity*. The latter relates to *linearity constraint* or *structural gradients*. Velocity gradients provide important cues about the 3-D layout of the visual scene. Formally, they can be described as *linear deformations* by a $2 \times 2$ velocity gradient tensor

$$\mathbf{T} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} = \begin{bmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{bmatrix} . \tag{3}$$

Hence, if $\boldsymbol{x} = (x, y)$ is a point in a spatial image domain, the linear properties of a motion field $\boldsymbol{v}(x, y) = (v_x, v_y)$ around the point $\boldsymbol{x}_0 = (x_0, y_0)$ can be characterized by a Taylor expansion, truncated at the first order:

$$\boldsymbol{v} = \bar{\boldsymbol{v}} + \bar{\mathbf{T}}\boldsymbol{x} \tag{4}$$

where $\bar{\boldsymbol{v}} = \boldsymbol{v}(x_0, y_0) = (\bar{v}_x, \bar{v}_y)$ and $\bar{\mathbf{T}} = \mathbf{T}|_{\boldsymbol{x}_0}$. By breaking down the tensor in its dyadic components, the motion field can be locally described through 2-D maps representing elementary flow components (EFCs):

$$\boldsymbol{v} = \boldsymbol{\alpha}^x \bar{v}_x + \boldsymbol{\alpha}^y \bar{v}_y + \boldsymbol{d}_x^x \left.\frac{\partial v_x}{\partial x}\right|_{\boldsymbol{x}_0} + \boldsymbol{d}_y^x \left.\frac{\partial v_x}{\partial y}\right|_{\boldsymbol{x}_0} + \boldsymbol{d}_x^y \left.\frac{\partial v_y}{\partial x}\right|_{\boldsymbol{x}_0} + \boldsymbol{d}_y^y \left.\frac{\partial v_y}{\partial y}\right|_{\boldsymbol{x}_0} \tag{5}$$

where $\boldsymbol{\alpha}^x : (x, y) \mapsto (1, 0)$, $\boldsymbol{\alpha}^y : (x, y) \mapsto (0, 1)$ are pure translations and $\boldsymbol{d}_x^x : (x, y) \mapsto (x, 0)$, $\boldsymbol{d}_y^x : (x, y) \mapsto (y, 0)$, $\boldsymbol{d}_x^y : (x, y) \mapsto (0, x)$, $\boldsymbol{d}_y^y : (x, y) \mapsto (0, y)$ represent cardinal deformations, basis of the linear deformation space.

It is worthy to note that the components of pure translations could be incorporated in the corresponding deformation components, thus obtaining generalized deformation components in which motion boundaries are shifted or totally absent. Although this does not affect the significance of the Taylor expansion in Eq. 5, the so-modified elementary components, present very different structural properties. Since a template-based approach cannot be used to extract single components, but only to perform pattern matching operations, the linear decomposition of the motion field has significance only for the definition of a proper representation space. Specific templates would be designed to optimally sample that representation space. In this work, we consider two different classes of deformation templates (opponent and non-opponent), each characterized by two gradient types (stretching and shearing), see Fig. 2. Due to their ability to detect the presence and the orientation of velocity gradients and kinetic boundaries, such cardinal EFCs and proper combinations of them resemble the characteristics of the cell in the Middle Temporal visual area (MT) [3] [4]. It is straightforward to derive that these MT-like components are well suited to provide the building blocks for the more complex receptive field properties encountered in the Medial Superior Temporal visual area (MST) [5] [6]:

$$\boldsymbol{v} = \boldsymbol{\alpha}^x \bar{v}_x + \boldsymbol{\alpha}^y \bar{v}_y + \frac{1}{2}(\boldsymbol{d}_x^x + \boldsymbol{d}_y^y)E + \frac{1}{2}(\boldsymbol{d}_x^x - \boldsymbol{d}_x^y)\omega + \frac{1}{2}(\boldsymbol{d}_x^x - \boldsymbol{d}_y^y)S_1 + \frac{1}{2}(\boldsymbol{d}_y^x + \boldsymbol{d}_x^y)S_2$$

where $E = (\bar{T}_{11} + \bar{T}_{22})/2$, $\omega = (\bar{T}_{12} - \bar{T}_{21})/2$, $S_1 = (\bar{T}_{11} - \bar{T}_{22})/2$, $S_2 = (\bar{T}_{12} + \bar{T}_{21})/2$ are the divergence, the curl and the two components of shear

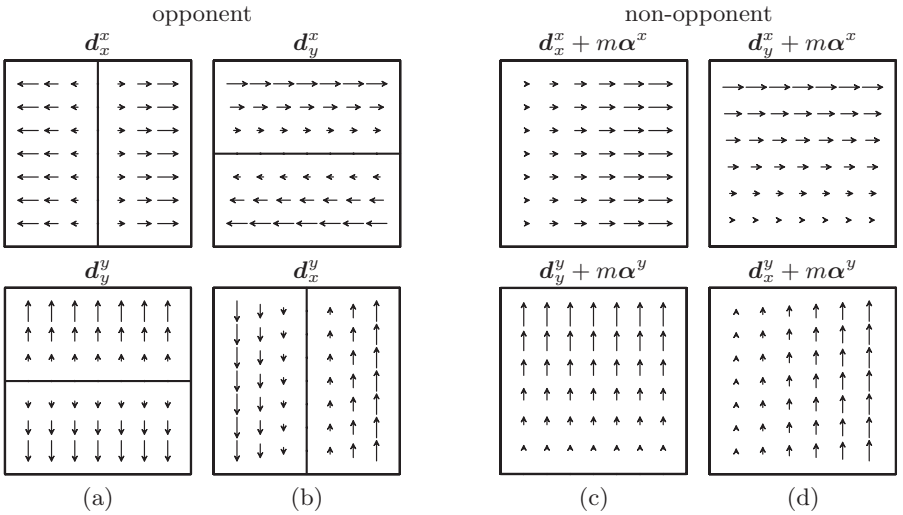



**Fig. 2.** Basic gradient type Gestalts considered. In stretching-type components (a,c) velocity varies *along* the direction of motion; in shearing-type components (b,d) velocity gradient is oriented *perpendicularly* to the direction of motion. Non-opponent patterns are obtained from the opponent ones by a linear combination of pure translations and cardinal deformations: $\boldsymbol{d}_j^i + m\boldsymbol{\alpha}^i$, where $m$ is a proper positive scalar constant

deformation, respectively (cf. [2]). These mixed EFCs constitute, together with the pure translations, an equivalent representation basis for the linear properties of the velocity field (see Fig. 3). Yet, they are rather complex since not only the speed, but also the direction of feature motion varies as a function of spatial position. Rigid body motion often generates simpler flow fields characterized by unidirectional patterns, as the cardinal EFCs considered in this study.

## 4 The Context Sensitive Filter

On the basis of the considerations presented in Section 2, the problem of evidencing the presence of a certain complex feature in the optic flow on the basis of both local and contextual information, is posed as an adaptive filtering problem. Local information act as the input *measurements* and the context acts as the *reference signal*, e.g., representing a specific motion Gestalt.

### 4.1 Measurement Equation

For each spatial position $(i, j)$ and at time step $k$, let us assume the optic flow $\tilde{\boldsymbol{v}}(i, j)[k]$ as the corrupted measure of the actual velocity field $\boldsymbol{v}(i, j)[k]$. For the sake of notation, we drop the spatial indices $(i, j)$ to indicate the vector

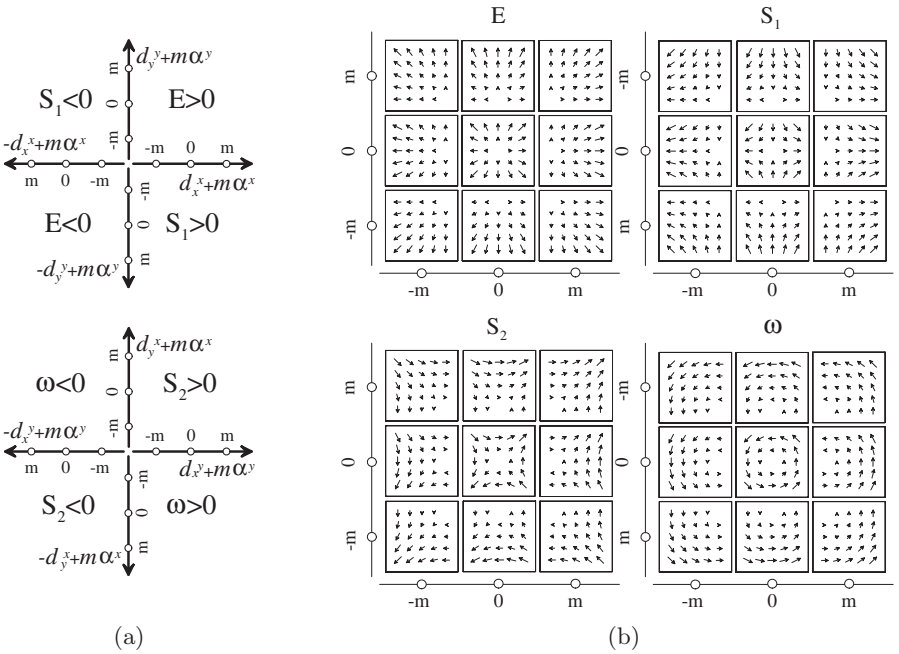



**Fig. 3.** (a) Two deformation subspaces obtained by the set of cardinal EFCs with different values of the parameter $m$. The quadrants of each subspace characterize an elementary deformation, as evidenced in (b) for expansion ($E > 0$), horizontal positive shear ($S_1 > 0$), oblique positive shear ($S_2$), and counterclockwise rotation ($\omega > 0$)

that represents the whole spatial distribution of a given variable. The difference between these two variables can be represented as a noise term $\boldsymbol{\varepsilon}(i,j)[k]$:

$$\tilde{\boldsymbol{v}}[k] = \boldsymbol{v}[k] + \boldsymbol{\varepsilon}[k] \ . \tag{6}$$

Due to the intrinsic noise of the nervous system, the neural representation of the optic flow $\mathbf{v}[k]$ can be expressed by a *measurement equation*:

$$\mathbf{v}[k] = \tilde{\boldsymbol{v}}[k] + \boldsymbol{n}_1[k] = \boldsymbol{v}[k] + \boldsymbol{\varepsilon}[k] + \boldsymbol{n}_1[k] \tag{7}$$

where $\boldsymbol{n}_1$ represents the uncertainty associated with a neuron's response. In this case the measurement matrix $\boldsymbol{C}$ is the identity operator. The approach can be straightforwardly generalized to consider indirect motion information, e.g., by the gradient equation [7] $-I_t[k] = \boldsymbol{\nabla}^T I[k]\tilde{\boldsymbol{v}}[k] + \boldsymbol{n}_1[k]$ where $\boldsymbol{\nabla}^T I$ and $I_t$ are the spatial image gradient and temporal derivative, respectively, of the image at a given spatial location and time. It is worthy to note that here the linear operator relating the quantity to be estimated to the measurement $I_t$ is *also* a measurement [8].

### 4.2 Process Equation

In the present case, the reference signal should reflect spatio-temporal structural regularities of the input optic flow. These structural regularities can be described statistically and/or geometrically. In any case, they can be defined by a process equation that models spatial relationships by the transition matrix $\boldsymbol{\Phi}$:

$$\boldsymbol{v}[k] = \boldsymbol{\Phi}[k, k-1]\boldsymbol{v}[k-1] + \boldsymbol{n}_2[k-1] + \boldsymbol{s} \ . \tag{8}$$

The state transition matrix $\boldsymbol{\Phi}$ is *de facto* a spatial interconnection matrix that implements a specific Gestalt rule (i.e., a specific EFC); $\boldsymbol{s}$ is a constant driving input; $\boldsymbol{n}_2$ represents the process uncertainty. The space spanned by the observations $\mathbf{v}[1]$, $\mathbf{v}[2]$,..., $\mathbf{v}[k-1]$ is denoted by $\boldsymbol{\mathcal{V}}_{k-1}$ and represents the internal noisy representation of the optic flow. We assume that both $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are independent, zero-mean and normally distributed: $\boldsymbol{n}_1[k] = N(0, \boldsymbol{\Lambda}_1)$ and $\boldsymbol{n}_2[k] = N(0, \boldsymbol{\Lambda}_2)$. More precisely, $\boldsymbol{\Phi}$ models space-invariant nearest-neighbor interactions within a finite region $\Omega$ in the $(i, j)$ plane that is bounded by a piecewise smooth contour. Interactions occur, separately for each component of the velocity vectors $(v_x, v_y)$, through anisotropic interconnection schemes:

$$\begin{aligned} v_{x/y}(i,j)[k] = {} & w_N^{x/y} v_{x/y}(i,j-1)[k-1] + w_S^{x/y} v_{x/y}(i,j+1)[k-1] + \\ & w_W^{x/y} v_{x/y}(i-1,j)[k-1] + w_E^{x/y} v_{x/y}(i+1,j)[k-1] + \\ & w_T^{x/y} v_{x/y}(i,j)[k-1] + n_1^{x/y}(i,j)[k-1] + s_{x/y}(i,j) \end{aligned} \tag{9}$$

where $(s_x, s_y)$ is a steady additional control input, which models the boundary conditions. In this way, the structural constraints necessary to model cardinal deformations are embedded in the lattice interconnection scheme of the process equation. The resulting lattice network has a *structuring effect* constrained by the boundary conditions that yields to structural equilibrium configurations, characterized by specific first-order EFCs. The resulting pattern depends on the anisotropy of the interaction scheme and on the boundary conditions. By example, considering, for the sake of simplicity, a rectangular domain $\Omega = [-L, L] \times [-L, L]$, the cardinal EFC $\boldsymbol{d}_x^x$ can be obtained through:

$$\begin{array}{ll} w_N^x = w_S^x = 0 & w_N^y = w_S^y = 0 \\ w_W^x = w_E^x = 0.5 & w_W^y = w_E^y = 0 \end{array} \quad s_x(i,j) = \begin{cases} -\lambda \text{ if } i = -L \\ \ \ \lambda \quad \text{if } i = L \\ \ \ 0 \text{ otherwise} \end{cases} \quad s_y(i,j) = 0$$

where the boundary value $\lambda$ controls the gradient slope. In a similar way we can obtain the other components. Given Eqs. (7) and (8), we may write the optimal filter for optic flow Gestalts. The filter allows to detect, in noisy flows, intrinsic correlations, as those related to EFCs, by checking, through spatial recurrent interactions, that the spatial context of the observed velocities conform to the Gestalt rules, embedded in $\boldsymbol{\Phi}$.

## 5 Results

To understand how the CSF works, we define the *a priori* state estimate at step $k$ given knowledge of the process at step $k-1$, $\hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_{k-1}]$, and the *a posteriori* state estimate at step $k$ given the measurement at the step $k$, $\hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_k]$. The aim of the CSF is to compute an *a posteriori* estimate by using an *a priori* estimate and a weighted difference between the current and the predicted measurement:

$$\hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_k] = \hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_{k-1}] + \boldsymbol{G}[k] \ \left(\boldsymbol{v}[k] - \hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_{k-1}]\right) \tag{10}$$

The difference term in Eq. (10) is the *innovation* $\boldsymbol{\alpha}[k]$ that takes into account the discrepancy between the current measurement $\boldsymbol{v}[k]$ and the predicted measurement $\hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_{k-1}]$. The matrix $\boldsymbol{G}[k]$ is the Kalman gain that minimizes the *a posteriori* error covariance:

$$\boldsymbol{K}[k] = E\left\{(\boldsymbol{v}[k] - \hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_k])(\boldsymbol{v}[k] - \hat{\boldsymbol{v}}[k|\boldsymbol{\mathcal{V}}_k])^T\right\} \ . \tag{11}$$

Eqs. 10 and 11 represent the mean and covariance expressions of the CSF output.

The covariance matrix $\boldsymbol{K}[k]$ provides us only information about the properties of convergence of the KF and not whether it converges to the correct values. Hence, we have to check the consistency between the innovation and the model (i.e., between observed and predicted values) in statistical terms. A measure of the reliability of the KF output is the Normalized Innovation Squared ($NIS$):

$$NIS_k = \boldsymbol{\alpha}^T[k] \ \boldsymbol{\Sigma}^{-1}[k] \ \boldsymbol{\alpha}[k] \tag{12}$$

where $\boldsymbol{\Sigma}$ is the covariance of the innovation. It is possible to exploit Eq. (12) to detect if the current observations are an instance of the model embedded in the KF [9]. Fig. 4 shows the responses of the CSF in the deformation subspaces ($E - S_1$, $\omega - S_2$) for two different input flows. Twentyfour EFC models have been used to span the deformation subspaces shown in Fig. 3a. The grey level in the CSF output maps represents the probability of a given Gestalt according to the $NIS$ criterion: lightest grey indicates the most probable Gestalt. Besides Gestalt detection, context information reduces the uncertainty on the measured velocities, as evidenced, for the circled vectors, by the Gaussian densities, plotted over the space of image velocity.

To assess the performance of the approach to obtain regularized patch-based motion estimation, we applied CSFs to optic flows of real-world driving sequences. Fig. 5 shows a road scene taken by a rear-view mirror of a moving car under an overtaking situations. A "classical" algorithm [10] has been used to extract the optic flow. Regularized motion estimation has been performed on overlapping local regions of the optic flow on the basis of the elementary flow components. In this way, we can compute a dense distribution of the local Gestalt probabilities for the overall optic flow. Thence, we obtain, according to the $NIS$ criterion, the most reliable (i.e. regularized) local velocity patterns, e.g., the patterns of local Gestalts that characterize the sequence (see Fig. 5).

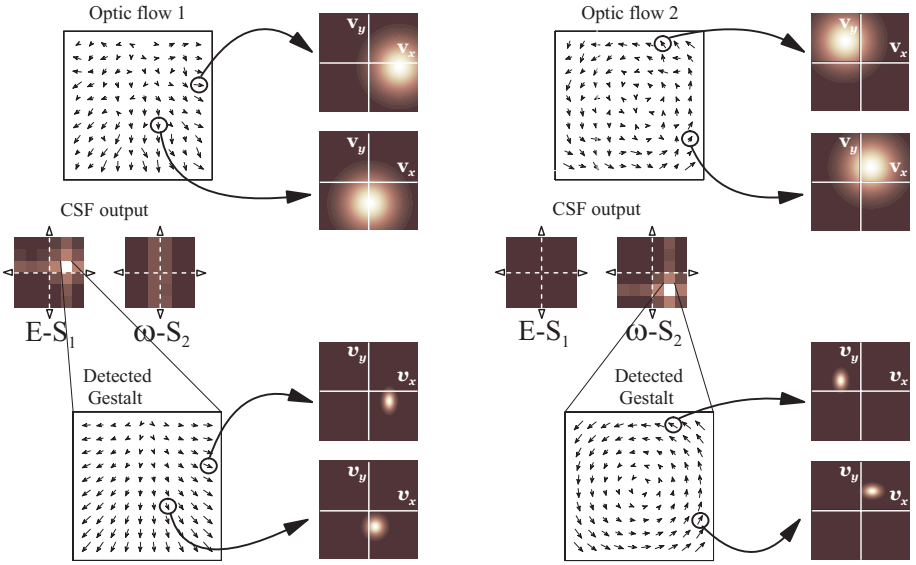
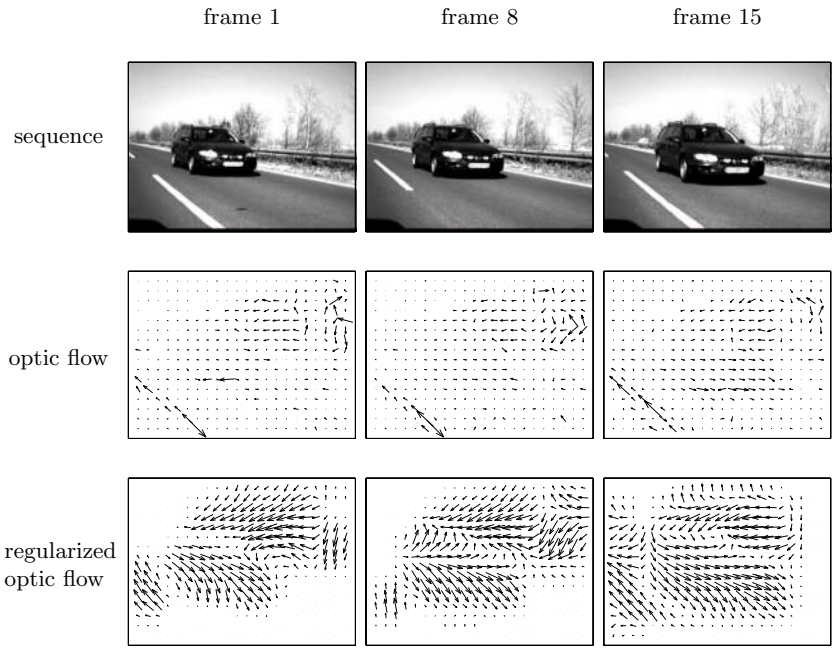



**Fig. 4.** Example of Gestalt detection in noisy flows

## 6 Discussion and Conclusions

Measured optic flow fields are always somewhat erroneous and/or ambiguous. First, we cannot compute the actual spatial or temporal derivatives, but only their estimates, which are corrupted by image noise. Second, optic flow is intrinsically an image-based measurement of the relative motion between the observer and the environment, but we are interested in estimating the actual motion field. However, real-world motion field patterns contain intrinsic properties that allow to define Gestalts as groups of pixels sharing the same motion property. By checking the presence of such Gestalts in optic flow fields we obtain context-based regularized patch motion estimation and make the interpretation of the optic flow more confident. We propose an optimal recurrent filter capable of evidencing motion Gestalts corresponding to 1st-order spatial derivatives or elementary flow components. A Gestalt emerges from a noisy flow as a solution of an iterative process of spatially interacting nodes that correlates the properties of the visual context with that of a structural model of the Gestalt. The CSF behaves as a template model. Yet, its specificity lies in the fact that the template character is not built by highly specific feed-forward connections, but emerges by stereotyped recurrent interactions (cf. the process equation). Furthermore, the approach can be straightforwardly extended to consider adaptive cross-modal templates (e.g, motion and stereo). By proper specification of the matrix $\mathbf{\Phi}$, the process equation can, indeed, potentially model any type of multimodal spatio-temporal relationships (i.e., multimodal spatio-temporal context). The presented approach can be compared with Bayesian inference and Markov



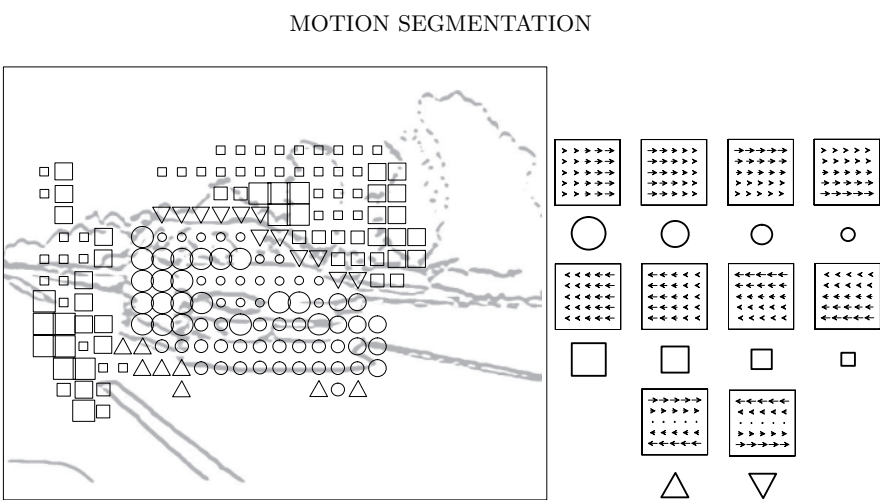




**Fig. 5.** Results on a driving sequence showing a road scene taken by a rear-view mirror of a moving car under an overtaking situations: Gestalt detection in noisy flows and the resulting motion segmentation (context information reduces the uncertainty on the measured velocities). Each symbol indicates a kind of EFC and its size represents the probability of the given EFC. The absence of symbols indicates that, for the considered region, the reliability of the segmentation is below a given threshold

Random Fields (MRFs). Concerning Bayesian inference, KF represents a recursive solution to an inverse problem of determing the distal stimulus based on the proximal stimulus, in case we assume: (1) a stochastic version of the regularization theory involving Bayes' rule, (2) Markovianity, (3) linearity and Gaussian normal densities. Concerning MRFs, they are used in visual/image processing to model context dependent entities such as image pixels and correlated features. If one assumes to have not the direct accessibility to the "system", we can refer to dynamic state space models [11] [12] [13] (cf. also Hidden MRF), given by the system's observations and an underlying stochastic process, which is included to describe the distribution of the observation process properly. In this perspective, Kalman's process equation can be related to a MRF. The presence of the measurement equation (observations) makes more evident the distinction between the feed-forward and feed-back components of our CSFs.

## Acknowledgements

We wish to thank G.M. Bisio and F. Wörgötter for stimulating discussions and M. Müelenberg of Hella for having provided the driving sequences. This work was partially supported by EU Project IST-2001-32114 "ECOVISION".

## References

[1] S. Haykin. *Adaptive Filter Theory.* Prentice-Hall International Editions, 1991. 387

[2] J. J. Koenderink. Optic flow. *Vision Res.*, 26(1):161–179, 1986. 388, 390

[3] V. L. Marcar, D. K. Xiao, S. E. Raiguel, H. Maes, and G. A. Orban. Processing of kinetically defined boundaries in the cortical motion area MT of the macaque monkey. *J. Neurophysiol.*, 74(3):1258–1270, 1995. 389

[4] S. Treue and R. A. Andersen. Neural responses to velocity gradients in macaque cortical area MT. *Visual Neuroscience*, 13:797–804, 1996. 389

[5] C. J. Duffy and R. H. Wurtz. Response of monkey MST neurons to optic flow stimuli with shifted centers of motion. *J. Neuroscience*, 15:5192–5208, 1995. 389

[6] M. Lappe, F. Bremmer, M. Pekel, A. Thiele, and K. P. Hoffmann. Optic flow processing in monkey STS: A theoretical and experimental approach. *J. Neuroscience*, 16:6265–6285, 1996. 389

[7] B. K. P. Horn and B. G. Schunck. Determinig optical flow. *Artificial Intelligence*, 17:185–204, 1981. 391

[8] E. P. Simoncelli. Bayesian multi-scale differential optical flow. In *Handbook of Computer Vision and Applications*, pages 297–322. Academic Press, 1999. 391

[9] Y. Bar-Shalom and X. R. Li. *Estimation and Tracking, Principles, Techniques, and Software.* Artech House, 1993. 393

[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. DARPA Image Understanding Workshop*, pages 121–130, 1981. 393

[11] A. C. Harvey. *Forecasting, structural time series models and the Kalman filter.* Cambridge University Press, Cambridge, 1989. 396

[12] M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer-Verlag, New York, 1997. 396

[13] H. R. Künsch. State space and hidden Markov models. In *Complex Stochastic Systems, no. 87 in Monographs on Statistics and Applied Probability*, pages 109–173. Chapman and Hall, London, 2001. 396

# Content Based Image Retrieval for Unsegmented Images

Marco Anelli[2], Alessandro Micarelli[1,2], and Enver Sangineto[1]

[1] Centro di Ricerca in Matematica Pura e Applicata
(CRMPA), Sezione "Roma Tre"
Via della Vasca Navale 79, 00146 Roma, Italia
[2] Dipartimento di Informatica e Automazione
AI Lab, Università degli Studi "Roma Tre"
Via della Vasca Navale 79, 00146 Roma, Italia
{anelli,micarel,sanginet}@dia.uniroma3.it

**Abstract.** We present a new method for image retrieval by shape similarity able to deal with real images with not uniform background and possible touching/occluding objects. First of all we perform a sketch-driven segmentation of the scene by means of a Deformation Tolerant version of the Generalized Hough Transform (DTGHT). Using the DT-GHT we select in the image some candidate segments to be matched with the user sketch. The candidate segments are then matched with the sketch checking the consistency of the corresponding shapes. Finally, background segments are used in order to inhibit the recognition process when they cannot be perceptually separated from the object.

## 1 Motivations and Goals

In this paper we present a new method for Content Based Image Retrieval (CBIR) based on the analysis of the object shapes. The availability of large image data bases makes necessary the development of efficient tools for retrieval of visual information perceptually *similar* to the user's requests. In [1, 2] we have presented an original CBIR technique based on a Deformation Tolerant version of the well known Generalized Hough Transform (DTGHT). In this paper we start from these works and we go further with the objective to augment the system precision. We introduce some segmentation rules corresponding to Gestalt principles (see, for example, [13] and [10]) exploiting the continuity properties of the edge segments and a more accurate matching method which compares the searched shape with the image lines. We have obtained an acceptance/rejection rule able to classify a data base image as *relevant* or *not relevant* with respect to the user's query.

The novelties of our approach with respect to similar systems is its capability to deal with every kind of real images, without caring of light conditions, with not uniform backgrounds and possible occluding objects. We refer to Section 4 for a comparison of our approach with existing techniques for image retrieval by shape similarity.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 398–409, 2003.
© Springer-Verlag Berlin Heidelberg 2003

The rest of the paper is organized as follows. In the next section we present the proposed methodology, while in Section 3 we show the obtained experimental results. In Section 4 we give an overview of other approaches to the problem and in Section 5 we conclude sketching a possible future direction of the work.

## 2 Line Segment Interpretation

We give here an overview of the method. Each image of the system data base is off-line pre-processed in order to extract an edge map and to reduce the noise (Section 2.1). On-line, the system accepts a user drawn query representing the shape she/he is interested to find in the data base (see Figure 3). The sketch is processed in order to extract a suitable representation (Section 2.2). Then, we compare the sketch representation with the previously processed images and we localize a region in the image which maximizes the likelihood of the presence of a shape similar to the sketch (Section 2.3). This phase is important because we do not manually select the interesting objects from their background like the most working CBIR systems presently do. Finally, we perform a verification phase composed of two distinct tests in order to confirm or reject the perceptual similarity (Section 2.4). In our trials, very few false positives have passed these tests despite the difficulty of the image domain adopted (see Section 3).

### 2.1 Pre-processing

In this section we explain the image processing phase, which is performed completely off-line when data base images are stored. In the first pre-processing step we apply the Crimmins filter [6] to the gray value image in order to reduce noise. Performing a few iterations with this filter we enhance the contour pixels weakening the texture ones. Afterwards we perform a standard edge extraction and thinning process using the Canny edge detector with Sobel $3 \times 3$ masks [4]. After this standard pre-processing, we use two *salience filters* to erase those thick textured areas which deteriorate the retrieval process. The two filters work in this way. The first one computes, for each edge pixel $p$, the edge density $d$ and the edge direction variance $\sigma$ in a $n_1 \times n_1$ squared area centered in $p$. If these two values exceed two given thresholds, $p$ is considered to be part of a thick texture area and so it is erased. The second filter computes, in a squared area $n_2 \times n_2$ centered on each pixel $p$, the number of edge pixels with the same direction value of $p$. If this number exceeds a threshold it means that $p$ is part of a regular textured area and we erase it. For further details we refer to [1, 2]. From now on, we indicate with $I$ the edge map of a generic image of the system data base after the filters' application. Figure 1 and Figure 2 respectively show a data base image example before and after the filters' applications.

The third step in the image processing phase exploits the property of continuity that stands between some pixels in $I$. We merge the pixels in $I$ with their neighbors in order to build line segments. We define a *segment* $s$ as an ordered list of points belonging to $I$: $s =< p_1, p_2, ...p_{n_s} >$ such that each $p_i$ $(i = 1, ..., n_s)$

**Fig. 1.** An example of data base image

**Fig. 2.** The image of Figure 1 after all the filters' application

is adjacent to $p_{i-1}$ and $p_{i+1}$ in a 8-connected interpretation of pixel adjacency. Moreover, $p_1$ and $p_{n_s}$ (the segment end-points) can alternatively be such that:

- they have only one adjacent point; or
- they are a junction with another segment, i. e., they have more than 2 adjacent points.

All the other points $p_i$ $(i = 2, ..., n_s - 1)$ have exactly 2 adjacent points. This segmentation process is performed because pixels that are part of the same continuous segment are more likely to belong to the same object. Segments have more semantic relevance than isolated pixels. We select from the image $I$ a set $Seg$ of segments whose length (number of adjacent points) is equal or greater then $2l + 1$. We refer to Section 2.3 for the parameter $l$ definition.

Finally, for each segment $s =< p_1, p_2, ...p_{n_s} >$ and each $p_j \in s$, we compute the pixel edge direction by the following:

$$\phi_I(p_j) = \frac{y_{j+10} - y_{j-10}}{x_{j+10} - x_{j-10}} \tag{1}$$

where $\phi_I(p)$ is the direction of the edge point $p$ and $(x_k, y_k)$ are the coordinates of the k-th point of $s$. Of course, attention must be paid for those points $p_j$

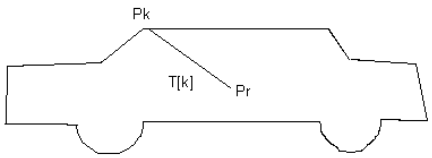



**Fig. 3.** An example of user sketch with its center of mass $p_r$. The vector $p_r - p_k$ is the k-th element of the R-Table

with $j \in [1, 9]$ or $j \in [n_s - 9, n_s]$. We compute $\phi_I$ by means of (1) because the edge directions obtained using Sobel masks (or similar masks) are not sufficently accurate for our purposes.

### 2.2 Sketch Processing

In this part we explain the representation of the user sketch ($S$). In order to simplify some technical steps concerning the matching between image segments and $S$, we assume that $S$ is composed of only an external closed contour. Nevertheless, this is not a limitation of the method, nor its extension to sketches with internal edges makes worse the whole computational cost. We have only chosen a technical simplification in order to speed up the working in progress.

First of all, we use $S$ as a model to build the R-Table $T$ [3]. Differently by the original Ballard's proposal, we do not index $T$ by means of the gradient direction of the points in $S$. Indeed, due to local dissimilarities between $S$ and a possible similar (but not identical) shape $S'$ in $I$, we usually expect that a point $p$ in $S$ and a corresponding point $p'$ in $S'$ are quite differently oriented. Thus, $T$ is built according to the following algorithm:

*Sketch Representation Construction(S)*
1 Compute the centroid $p_r$ of $S$.
2 For each point $p_k$ $(k = 1, ..., m)$ of $S$,set:
$T[k] := p_r - p_k$ and
$\phi_S[k] := \frac{y_{k+10} - y_{k-10}}{x_{k+10} - x_{k-10}}$

where $\phi_S$ is the analogous of $\phi_I$ for the direction of the points in $S$ and $m = |S|$ ($m$ is the cardinality of $S$). Figure 3 shows a user sketch and an element $T[k]$ of the R-Table.

### 2.3 Object Localization

In this section we want to find the most probable position for the user sketch $S$ in $I$. To do this we have used and modified the method presented in [1] where we have shown a Deformation Tolerant version of the Generalized Hough Transform

(DTGHT) that can be used for image retrieval purposes. In fact, the original Ballard's Generalized Hough Transform (GHT) is a powerful method that allows us to locate an object instance in an image possibly containing not uniform background and/or occluding objects [3]. Unfortunately, the original GHT can detect only objects that exactly match the template (it is an *object identification* technique). This is a strong limit for Image retrieval. In order to use the GHT properties, we need to modify its voting mechanism allowing a tolerance to shape's not-rigid deformations. In the following we briefly explain the method presented in [1] together with some modifications regarding edge orientations and other adaptations to our purposes, referring the reader to the original paper [1] for more details.

In the voting phase we increment all those accumulator's positions belonging to a bi-dimensional squared range centered in $p_r$ (called the *voting window* $W(p_r)$), for each edge point $p$ of $I$ and each vector $v = p_r - p$ in $T$. This is efficiently achieved using two accumulators $A$ and $B$, both of the same dimensions of the image $I$ and splitting the voting process in two sequential phases: the *exact voting phase* and the *spreading phase*. The spreading phase is realized by means of a local filter operation on $B$ performed using a dynamic programming technique (for details, see [1]):

*Voting Procedure*$(Seg, T, \phi_I, \phi_S)$
$**$ *exact voting phase:*
1 The accumulators $A$ and $B$ are set to 0.
2 For each $s \in Seg$ and each $p \in s$ $(p = (x, y))$, do:
3 For each $i$ $(i = 1, ..., m)$ do:
4 Let $T[i] = (x_i, y_i)$.
If $\phi_S(i) - \alpha \leq \phi_I(p) \leq \phi_S(i) + \alpha$
then $A[x + x_i, y + y_i] := A[x + x_i, y + y_i] + 1$.
$**$ *spreading phase:*
5 For each point $(x, y)$ of $B$ do:
6 For each $(x_1, y_1)$ s.t. $|x - x_1| <= l$, $|y - y_1| <= l$, do:
7 $B[x, y] := B[x, y] + A[x_1, y_1]$.

In Step 3, $\alpha$ is a threshold used to compare the directions of corresponding pixels in $I$ and $S$ (presently, $\alpha = \pi/8$). Moreover, the parameter $l$ determines the voting window size, whose side is $2l + 1$. The side length of the voting window is also used in Section 2.1 as a threshold to reject too small segments. Finally, we fix $l = 0.03d$, where $d$ is the diagonal of the $S$' minimum enclosing rectangle, in order to vary $l$ according to the sketch size.

If $M$ is the point in $B$ with the highest score after the above mentioned algorithm application, then we take it as the most likely position for the center of mass of a possible shape similar to $S$ (see Figure 4). In other words, if such a shape exists in $I$, $M$ is its most likely centroid. Experimental results presented in [1] confirm this assumption. Nevertheless, differently from the original GHT, in which the score of the accumulator in its maximum is equivalent to the number of points of $S$ in $I$, with a DTGHT approach, due to the augmented voting

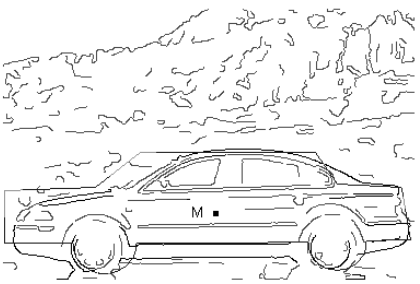



**Fig. 4.** The sketch of Figure 3 supermposed on Figure 2. Its centroid $M$ corresponds to the position with the highest score in the $B$ accumulator (see the test for details)

area, it is sometimes possible that a thick and random distribution of points and segments in $I$ can produce a high value $B[M]$ not corresponding to a shape really similar to $S$. In order to reject such cases, in the next section we perform some tests on the segments distribution around $M$. The segments are chosen taking into account $M$ and the R-Table $T$, hence we exploit the DTGHT for a model driven segmentation of the scene in $I$ (where the model is the sketch).

### 2.4 Verification Phase

In the following we assume that $M$ is the point in $I$ found as described in Section 2.3, $m = |S|$ and $W$ is the voting window. The aim of the verification phase we are going to explain in this section is to verify if the disposition of the segments in $I$ around $M$ is really perceptually similar to the $S$' shape. In order to avoid false positives, first of all we check the portion of $S$ which can be matched with segments in $I$. This is intuitively done by projecting $S$ on $I$ (let us call $S'$ such projection, with $M$ its centroid), looking for segments on $I$ in the the neighborhood of $S'$ and marking the portion of $S$ *justified* by these segments (see below, the algorithm *Justification test*).

Furthermore, we need to take into account *inhibitory* segments intersecting the neighborhood of $S'$ but not really following the shape $S$. For example, if you look to Figure 5, you can see that the presence of a high number of segments following the shape of $S'$ (the projection of a sketch representing a pistol) is not perceptually meaningful if they cannot be distinguished by other adjacent segments not following $S'$. We deal with this problem with the algorithm *Inhibitory test*.

In the following we give the operational definitions for justifying and inhibitory segments.

A point $p$ belonging to $I$ is called a *valid* point if:

$$\exists i \in [1, m] : p + T[i] \in W(M) \wedge |\phi_I(p) - \phi_S(i)| \leq \alpha, \tag{2}$$

**Fig. 5.** The sketch of a pistol superimposed to image 2 in the most likely position suggested by the DTGHT application (see Section 2.3). Even though a lot of horizontal segments are near to the pistol projection and could *justify* it (Section 2.4), the different disposition of the other segments can be exploited by the system to *inhibit* the pistol recognition (Section 2.4)

where $\alpha$ is the threshold introduced in Section 2.3 and $i$ is called a *valid hypothesis* for $p$. We call a segment $s_i$ a *justifying* segment if:

$$|V_i| \geq k_1 |s_i|, \tag{3}$$

where $k_1$ is a fixed threshold (presently $k_1 = 0.7$) and $V_i$ is the set of all the valid points of the segment $s_i$. Let $J$ be the subset of $Seg$ of all the justyfing segments. Figure 6 (top) shows the set $J$ for the image of Figure 2 and the sketch of Figure 3.

Finally, a segment $s_i$ is called an *inhibitory* segment if $s_i$ is not a justifying segment and:

$$k_2' |s_i| \geq |F_i| \geq k_2 |s_i|, \tag{4}$$

where $k_2$ and $k_2'$ are two fixed thresholds (presently $k_2 = 0.2$, $k_2' = 0.4$) and $F_i$ is the set of all the points $p$ in $s_i$ which satisfy:

$$\exists i \in [1, m] : p + T[i] \in W(M). \tag{5}$$

Let $H$ be the subset of $Seg$ of all the inhibitory segments. The verification test is given by the logical and of the justification test and the inhibitory test.

*Justification Test(J)*
1 For $k \in [1, m]$, set the local boolean vector $B[k] := false$.
2 for each $s_i \in J$ do:
3 for each point $p \in V_i$ do:
4 for each valid hypothesis $k$, set $B[k] := true$.
5 If $|\{B[k] : B[k] = true\}| \geq k_3 m$
then return true,
else return false.

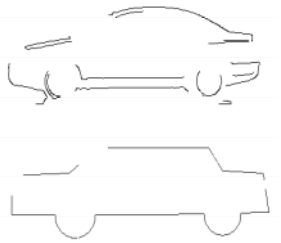


**Fig. 6.** Justification of a shape. Top: the justification segments for the image of Figure 2 and the sketch of Figure 3. Bottom: the justified portion of the sketch. In this case, $|\{B[k] : B[k] = true\}| = 0.89\ m$

In the above algorithm $k_3$ is a threshold, currently fixed to 0.7. We check in this way that at least a portion $k_3$ of $S$ is present in $I$ in the right position with respect to $M$. This test avoids the possibility of thick areas in $I$ producing "false" maxima in which the vote is not distributed along a sufficient portion of the $S$' shape. Figure 6 (bottom) shows the portion of $S$ justified by the Justification Test algorithm for the image of Figure 2 and the sketch of Figure 3.

Finally, the following is the inhibitory test:

*Inhibitory Test(H,J)*
1 If $\Sigma_{s_i \in H}|s_i| < \Sigma_{s_i \in J}|s_i|$
then return true,
else return false.

Before concluding this section, we observe that, in order to cope rotation and scale changing factors, we have to iterate algorithms of Section 2.3 and 2.4 for discrete values of rotation and scale. This situation is the same of the original Generalized Hough Transform [3]. Nevertheless, it is worth noting that the deformation tolerance allows invariance for scale differences coped by the voting window, and, for foreground searched objects, few iterations are enough to analyze almost all the interesting situations. Presently, in our experimental trials only 5 iterations have been enough for retrieval of the most important objects represented in the repository images.

Concerning rotation, up to now our system does not cope rotation invariance issues. However, in an image retrieval application, we can suppose the user usually draws its sketch with the right object orientation (an "horizontal" car and so on). For this reason we believe rotation invariance issues are less important with respect to other rigid and not rigid transformation invariance completely coped by the proposed system [5].

## 3 Experimental Results

We have implemented our method in Java 1.3 and tested it on a Pentium IV, 1.4 GHz, 1 GB RAM. All our trials (a few hundreds) have been executed in less than two seconds per image (1.47 seconds on average) with images from $200 \times 200$ up to $380 \times 350$ pixels. All the above mentioned times include scale invariance iterations.

The system data base is composed of 283 images randomly taken from the Web. They show a great variety of subjects: 26 cars, 22 horses, 20 guitars, 20 pistols, 20 tennis rackets, 20 saxophones, 20 bottles, 20 watches, 15 crucifixes, 10 vases. Moreover, the data base contains different other subjects such as: animals, faces, landscapes, airplanes, boats, fruits, mushrooms, various cups and so on.

No simplifying assumption has been done about images: they usually have not uniform backgrounds and often the retrieved objects are occluded by other objects. No manual segmentation has been performed on the images.

We paid attention to choose as users people not aware of the data base images' shapes. Indeed, only one of the three users in Table 1 (User 3) has had the possibility to observe the repository images before his drawings; nevertheless, his scores are not far from the others. Each user has performed six queries. The drawn subjects are: a car, a pistol, a tennis racket, a watch, a bottle and a guitar. We report our results in Table 1 and Table 2. Figure 7 shows some examples of user drawn sketches. As you can see, they are not *ad hoc* shapes realized taken into account the data base real images but very stylyzed pictures. Finally, Figure 8 shows some examples of retrieved items for the sketch of a car.

## 4 Related Works

There are a lot of methods and techniques proposed in literature for image retrieval by shape similarity. Most of them can be classified in three main categories: *statistic techniques*, *deformable template matching* and *multiscale representations* (see, for example, [12]). It is interesting noting that almost all these approaches need to work with pre-segmented images in which the interesting

**Table 1.** Precision and recall scores for the three users. The first column shows the type of the object drawn by each user. The other columns show, for each user, respectively the precision and recall values obtained for each query

| Object type | User 1 | | User 2 | | User 3 | |
|---|---|---|---|---|---|---|
| Car | 0.94 | 0.65 | 0.81 | 1 | 0.91 | 0.81 |
| Pistol | 1 | 0.70 | 0.93 | 0.70 | 0.86 | 0.60 |
| Tennis Racket | 0.90 | 0.50 | 0.73 | 0.85 | 1 | 0.95 |
| Watch | 0.28 | 0.10 | 0.66 | 0.30 | 0.75 | 0.15 |
| Bottle | 0.60 | 0.45 | 0.75 | 0.60 | 0.93 | 0.7 |
| Guitar | 0.93 | 0.75 | 0.94 | 0.80 | 0.66 | 0.50 |

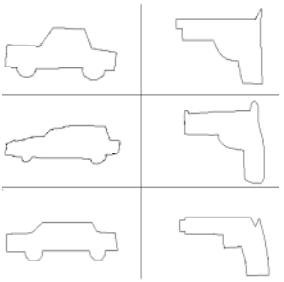
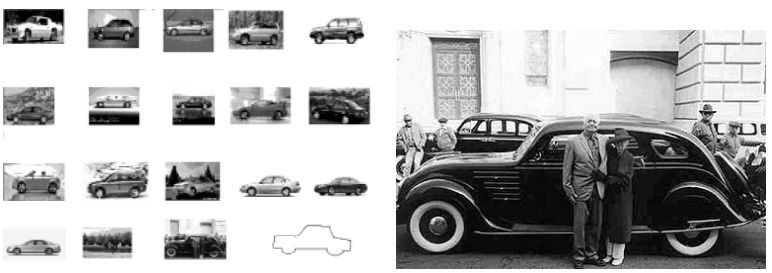


**Fig. 7.** Some examples of user sketches for a car an a pistol. The three rows (each row a couple of drawings) have been drawn respectively by the first, the second and the third users of Table 1



(a) (b)

**Fig. 8.** (a) The 18 images retrieved by the system according to the sketch of a car drawn by the first user of Table 1 (in the bottom-right corner). The second image on the last row is a false positive. (b) The last retrieved image for the same query, showing a car with a large occlusion (two human beings)

objects have been previously (manually) selected from their background or from other touching/occluding objects. For example, statistic methods [8], based on a feature vector representation of a fixed number of global features, extract such features from the objects' silhouettes (either their boarder or their internal pixels). But a silhouette will change its global features if merged with other objects.

Also multiscale representations usually implicitly or explicitly assume that the image is composed of an isolated object silhouette from which it is possible to extract the external contour. From the contour they extract the curvilinear abscissa which is then progressively smoothed performing a convolution with a gaussian curve whose variance is related with the scale factor [11].

Finally, deformable template matching techniques are usually able to work with not uniform backgrounds and/or occluding objects. Nevertheless, the interesting objects in each image have to be selected in order to initialize the

**Table 2.** Average values computed for each query category on the number of users (3). The second column shows the number of relevant items in the data base, Columns 3-7 the system's average results

| Query | Relevant | Correctly retrieved | Falsely retrieved | Missed | Precision | Recall |
|---|---|---|---|---|---|---|
| Car | 26 | 21.33 | 3 | 4,67 | 0.88 | 0.82 |
| Pistol | 20 | 13.30 | 1 | 6.70 | 0.93 | 0.74 |
| Racket | 20 | 15.33 | 2.33 | 4.67 | 0.86 | 0.76 |
| Watch | 20 | 3.66 | 3 | 16.34 | 0.54 | 0.18 |
| Bottle | 20 | 11.67 | 3.66 | 8.33 | 0.76 | 0.58 |
| Guitar | 20 | 13.67 | 2.33 | 6.33 | 0.85 | 0.68 |
| Average | 21 | 15.02 | 2.50 | 7.80 | 0.80 | 0.62 |

first sketch superimposition from which the iterative deformation process will start [7]. Otherwise, the sketch initialization parameters can be obtained by means of an exhaustive exploration of the image, as in [9] in which the authors iteratively change the rotation and translation parameters of the initial sketch superimosition. Unfortunately, deformable template matching techniques often have expensive computational costs and iterative initialization can bring to long execution times [9].

The necessity of almost all image retrieval by shape similarity systems of images with isolated objects has brought to the creation of CBIR benchmarks composed of only object silhouettes in uniform background, such as the benchmarks proposed by the MPEG-7 group. However, we believe that images with isolated objects are not very common in every-day domain-independent applications. We think that one of the most important challange for CBIR systems is the possibility to work with real images, with not uniform background and possible touching/occluding objects.

The work proposed in this paper exploits the capabilities of the GHT [3] and the DTGHT [1, 2] to work, respectively, with unsegmented images and with inexact matching and adds the analysis of continuous line segments in order to augment the system precision. We now have a rejection rule and a precision rate that is very high with respect to the difficulty of the image domain.

## 5 Conclusions and Future Directions

In this paper we have presented a CBIR method based on two main principles. First of all we use the DTGHT, a version of the GHT for inexact matching, in order to locate the most likely position of the sketch in the image and select line segments candidate for the image-sketch matching. The DTGHT performs a model driven image segmentation, where the model is the sketch. Afterwards, the selected segments are compared both with the sketch shape checking for their compatibility and with the surrounding background to inhibit false matching.

Continuity properties of line segments are exploited in all the method phases, in the pre-processing step, in the voting mechanism and in the final verification.

In this way we define shape information by means of two levels of representation: points and segments, both used by the matching process.

We have tested our system in a real final-user situation, with users not aware of the system data base and with a difficult image domain composed of images with not uniform background and occluding objects. Computational times and retrieval results are satisfactory and we are not aware of working systems with the same performances in the same domain. We plan a further experimentation with some of the MPEG-7 benchmarks although in this way some potentialities of our system cannot be well exploited.

## References


[1] M. Anelli, A. Micarelli, and E. Sangineto. A deformation tolerant version of the generalized hough transform for image retrieval. In *Fifteenth European Conference on Artificial Intelligence (ECAI 2002), Lyon, France*, 2002. 398, 399, 401, 402, 408

[2] M. Anelli, A. Micarelli, and E. Sangineto. A new content based image retrieval method based on a sketch-driven interpretation of line segments. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico*, 2003. 398, 399, 408

[3] D. H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13. No. 2:111–122, 1981. 401, 402, 405, 408

[4] J. Canny. A computational approach to edge detection. *IEEE Trans. on PAMI*, 8. No. 6:679–698, 1986. 399

[5] C. Colombo and A. Del Bimbo. Visible image retrieval. *Image Data Bases - Search and Retrieval of Digital Imagery. Ed. L. Bergman and V. Castelli, Wiley*, pages 11–33, Chapter 2, 2002. 405

[6] T. Crimmins. The geometric filter for speckle reduction. *Applied optics*, 24. No. 10, 1985. 399

[7] A Del Bimbo and P Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Trans. on PAMI*, 19. No. 2:121–132, 1997. 408

[8] M. Flickner et al. Query by image and video content: the QBIC system. *IEEE Computer*, 28. No. 9:23–32, 1995. 407

[9] A. K. Jain, Y. Zhong, and S. Lakshmanan. Object matching using deformable templates. *IEEE Trans. on PAMI*, 18:267–278, 1996. 408

[10] K. Koffka. *Principles of Gestalt Psychology.* New York: Harcourt, Brace and World, 1935. 398

[11] F Mokhtarian. Silhouette-based isolated object recognition through curvature scale-space. *IEEE Trans. on PAMI*, 17. No 5:539–544, 1995. 407

[12] A W M Smeulders, M Worring, S Santini, A Gupta, and R Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. on PAMI*, 22. No 12:1349–1380, 2000. 406

[13] S. Ullman. *High–level Vision. Object Recognition and Visual Cognition.* A Bradford Book. The MIT Press Cambridge, Massachusetts, 1996. 398

# A Mechanism of Coalition Formation in the Metaphor of Politics Multiagent Architecture

Antonio Chella, Rosario Sorbello, Daniele Ribaudo, Ignazio V. Finazzo, and Leonardo Papuzza

Dipartimento Ingegneria Informatica, ICAR-CNR
Viale delle Scienze, 90128 Palermo, Italy
chella@unipa.it, rosario@csai.unipa.it
http://www.csai.unipa.it/

**Abstract.** Hybrid Multi-Agent Architectures allow the support of mobile robots colonies moving in dynamic, not predictable and time variable environments in order to achieve distributed solving strategies that develop collective team-oriented behaviors for solving complicate and difficult tasks. The development of a new robotic architecture for the coordination of a robot colonies in dangerous, unknown and dynamic environment is outlined. The name of this new architecture is Metaphor of Politics (MP), because it loosely takes inspiration from the political organizations of democratic governments.

## 1 Introduction

A new hybrid and dynamic architecture to coordinate a robots team for complex tasks in dynamic, not predictable environments is proposed. The architecture [9, 10] takes inspiration from the political organizations of the democratic governments. The main idea behind the architecture is that the leadership is not owned by a single robot, but instead by a government of robots. A second group of robots are the "robot citizens". In this way we reach a compromise among the centralized and the distributed approaches. The goal is to have a decentralization of the planning actions, where each robot saves a deliberative independence status without losing its own reactivity. The agents receive high level goals by the government members and exploits their own deliberative capabilities to choose the faster strategy. The idea is to coordinate a colony of robots that are basically smart and able to exhibit complex behaviors in order to accomplish high level target but at the same time these architecture proposes a mechanism to make a new coalition caused by the failure of the government strategy and by a general inefficiency of the whole colony during the reaching of the mission targets.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 410–422, 2003.
© Springer-Verlag Berlin Heidelberg 2003

## 2 Mathematical Model

The MP architecture considers a colony composed of $H$ robot and $M$ political parties, with $M \leq H$, which guarantees the presence at least of one robot for every party. In our architecture we associate to every robot a set of issues which express individual features, as attitude to the risk, reactive or deliberative abilities, abilities in the exploration and in the objects recovering. Every issue can have a limited set of values: 0 (don't care), -1 (absolutely not) and 1 (absolutely yes). Every party is identified by a group of robots, so it can be represented by an ideal robot. Every robot is identified by N features; for every robot $i$ and party $j$ there is a vector of $n$ issues: $I_i^R$,$I_j^P \in M^{(n\times 1)}$ where $i = 1 \ldots H$, $j = 1 \ldots M$, $P$ = party and $R$ = robot. As example to describe our model, we consider 3 issues which we identify with the following terms and meanings:

- WELFARE: *Energy of the robot*
- DEFENSE: *Attitude to the risk*
- LABOR: *Amount of work*

Every issue is weighted through a non negative coefficient (from 0 to + $\infty$); the coefficient is the intensity or the strength of the issue. Every robot $R_i$ and party $P_j$ will be represented by a vector with $n$ components as following showed:

$$R_i = S_i^R \cdot I_i^R \ , \quad P_j = S_j^P \cdot I_j^P \tag{1}$$

where $S_i^R$, $S_j^P$ are diagonal $n \times n$ matrixes containing the weights of the robot issues and of the parties issues; $R_i$ and $P_j$ are representation of a robot and of a party in a multi-dimensional space called ROBOT ISSUES SPACE.
The following example shows a situation of a colony made up 11 robots and 3 parties. Let us consider the following vectors of issues:

$$I_1 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \ I_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \ I_3 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \ , \tag{2}$$

and the matrixes of weights:

$$W_1 = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 0 \end{bmatrix} \ W_2 = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \\ 0 & 0 & 20 \end{bmatrix} \ W_3 = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3}$$

the political parties will be described by the following vectors:

$$P_1 = \begin{bmatrix} 40 \\ -20 \\ 0 \end{bmatrix} \ P_2 = \begin{bmatrix} 20 \\ 20 \\ 20 \end{bmatrix} \ P_3 = \begin{bmatrix} -20 \\ 40 \\ 0 \end{bmatrix} \ . \tag{4}$$

The values of the components for each robots are random between 0 and 100:

$$R_1 = \begin{bmatrix} 29.74 \\ -4.91 \\ 0.00 \end{bmatrix} \quad R_2 = \begin{bmatrix} 65.01 \\ -98.29 \\ 0.00 \end{bmatrix} \quad R_3 = \begin{bmatrix} 40.00 \\ -19.87 \\ 0.00 \end{bmatrix} \quad R_4 = \begin{bmatrix} 73.33 \\ -37.58 \\ 0.00 \end{bmatrix}$$

$$R_5 = \begin{bmatrix} 41.98 \\ 75.36 \\ 79.38 \end{bmatrix} \quad R_6 = \begin{bmatrix} 91.99 \\ 84.47 \\ 36.77 \end{bmatrix} \quad R_7 = \begin{bmatrix} 62.08 \\ 73.12 \\ 19.38 \end{bmatrix} \quad R_8 = \begin{bmatrix} -90.48 \\ 56.92 \\ 0.00 \end{bmatrix} \tag{5}$$

$$R_9 = \begin{bmatrix} -23.44 \\ 54.87 \\ 0.00 \end{bmatrix} \quad R_{10} = \begin{bmatrix} -33.51 \\ 65.55 \\ 0.00 \end{bmatrix} \quad R_{11} = \begin{bmatrix} -62.73 \\ 69.90 \\ 0.00 \end{bmatrix}$$

In figure 1(a) it is showed the graphic representation of the robots and parties in the 3D space. The heterogeneity of the robots inside the colony is described by with a Roles Matrix ($RM$) which shows the capability for a robot to cover a role in the government; having $H$ robot and 4 roles the matrix will have an $H \times 4$ dimension. Referring to the numeric example, the roles matrix can be considered as follows:

$$\mathbf{RM} = \begin{bmatrix} 0\,1\,1\,1\,0\,1\,1\,0\,1\,1\,1 \\ 1\,0\,1\,1\,1\,0\,1\,1\,0\,1\,1 \\ 1\,1\,0\,1\,1\,1\,0\,1\,1\,0\,1 \\ 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix} \begin{matrix} \leftarrow 1^{st}\ \mathit{Role} \\ \leftarrow 2^{nd}\ \mathit{Role} \\ \leftarrow 3^{rd}\ \mathit{Role} \\ \leftarrow 4^{th}\ \mathit{Role} \end{matrix}$$

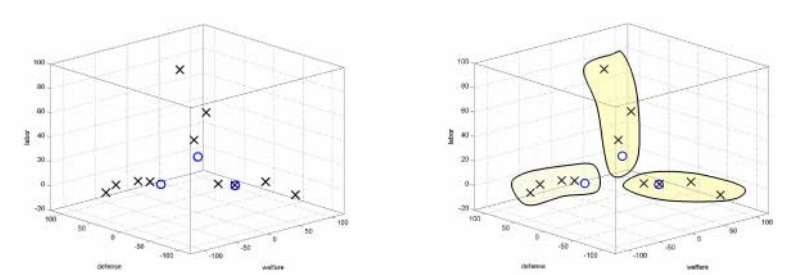
### 2.1 Voting Process

The voting process is divided in two steps. The first step is *Cluster Identification*; the classical clusterizzation techniques can be applied to our problem for the identification of the membership's groups; the literature provides several criteria; in particular we focus on *Voronoi tessellation* [17].

In the MP architecture, the cluster identification groups the robots of the colony on the basis of the membership's party; this choice is based on the consideration that every robot has a political orientation depending on the closest political party; a robot $R_i$ will belong to the $P_j$ party if the following condition is verified:

$$R_i \in P_j \Leftrightarrow d_{i,j} = \min_k \{d_{i,k}\} \quad k = 1, 2, \ldots M \tag{6}$$

with $d_{i,k}$ the euclidean distance among the robot $R_i$ and the generic party $P_k$ in the ROBOT ISSUES SPACE. The figure 1(b) shows the result of the clustering process. The first step is the *Vote Extraction*: the vote expression by a robot is simulated by a random extraction of a number included in interval [0,1], divided in its turn into $M$ sub-intervals, each one associated to the $M$ parties. The extraction provides a value that we associate to the vote expression by the robot for the considered party. For every robot $i$ of the colony the subdivision of



(a) Robots and parties (b) Clustering formation

**Fig. 1.** Graphic representation of robots, parties and clustering formation

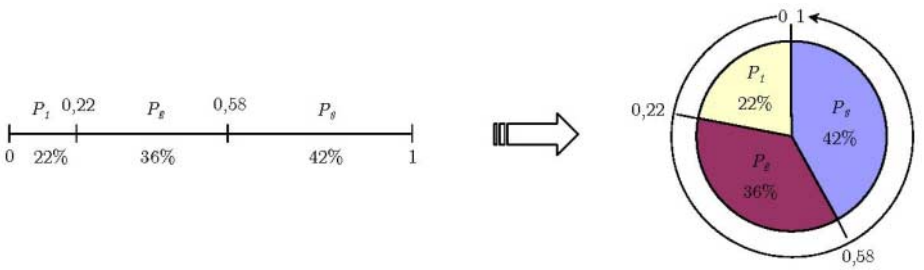



**Fig. 2.** Division of the [0,1] interval into 3 regions for the robot $R_i$

the interval [0,1] is made through the relative distances of the robot from each party $j$:

$$d_{i,j}^{REL} = \frac{\sum_{k \neq j} d_{i,k}}{\sum_k d_{i,k}} \% \qquad k = 1, 2, \ldots M; \tag{7}$$

To obtain $\sum d_{i,j}^{REL} = 100\%$ for each robot $i$, we normalize by a value equal to $1/(M-1)$. The figure 2 shows the division of the [0,1] interval for the robot $R_1$. This representation allows us to analyze the voting mechanism by a *Montecarlo* random extraction.

### 2.2 Coalition Formation

The formation of the political coalition which constitutes the new government is made with the support of a linear space, the POLITICAL IDEOLOGY SPACE. This space represents all the robots and the parties belonging to the ROBOT ISSUES SPACE. The mapping between the two spaces is performed by a mapping function $f(\cdot)$ which makes a scaling among the two spaces of representation and it groups the robots showing the same vote during the voting process.

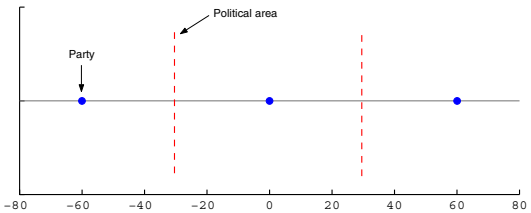



**Fig. 3.** Scaling of parties

To execute a scaling with a correct meaning, we make the following considerations: in analogy to a parliament the axe of coalitions is generally divided, into 3 big strips which represent the ideologies of the left, center and right political trends. The origin of the linear space is placed in such a way to coincide with the pure political center, in order to associate negative values to the left parties and positive values to the right ones. Two functions, $f_R(\cdot)$ and $f_L(\cdot)$, are introduced in order to highlight, if applied to a $P_j$ party, the aspects which respectively characterize the right and left trend through a scalar value; the considered party will have a political trend given by a better compromise among right and left one. If we represent the position of the party $P_j$ with $p_j$ in the space of the coalitions, we consider the following relation:

$$p_j = f(P_j) = f_R(P_j) - f_L(P_j) \ . \tag{8}$$

A positive value identifies a right party while a negative one identifies a left party; values closer to zero identify right-center or left-center parties according to the sign. The above mentioned functions can be assumed linear or non linear; in the first case the functions are simply some vectors of coefficients which opportunely weight the several members of the party; the value to be given to these coefficients is related to the meaning associated to the issues of the parties in order to identify the ideology in the ROBOT ISSUES SPACE. In the case of linear $f$, the function of scaling is:

$$p_j = M_R^T \cdot P_j - M_L^T \cdot P_j = (M_R - M_L)^T \cdot P_j \tag{9}$$

where $M_R, M_L \in M^{(n\times 1)}$. We introduce a scale factor for the positioning of the parties to avoid a dispersion of the parties themselves, keeping unchanged the relative distances. Referring to the previous example, assigning to matrixes $M_R$ and $M_L$ the following values:

$$M_R = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \ M_L = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \ , \tag{10}$$

We obtain the representation showed in figure 3.

The scaling of the single robots is made in a recursive way with respect to the scaling of the parties which have been previously voted: a first positioning is identified through the vote expressed; a robot which voted for a $P_j$ party

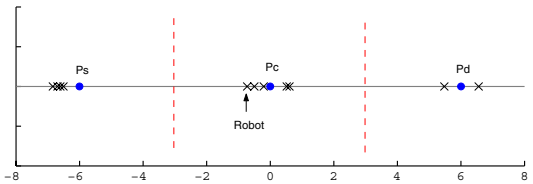



**Fig. 4.** Scaling of the robots

will be indicatively located in a region around $p_j$ in the space of the coalitions; a more refined positioning allows to place the robot to the right or to the left of the voted party. The scaling of robots is performed using an useful mapping function. Referring to the previous example and considering a scale factor equal to $\frac{1}{10}$, the robot scaling gives the graphic representation of figure 4.

To every robot $i$ is associated a political mass $m_{i,j}$ which represents the weight inside the voted party $j$; the calculation of such a mass takes place by a relation analogous to the calculation of the relative distances:

$$m_{i,j} = \frac{\sum_{k \neq i} d_{k,j}}{\sum_k d_{k,j}} \qquad \text{if } i \text{ voted } j \text{ , 0 otherwise} \tag{11}$$

where the index $k$ describes all the robots of the colony which expressed a vote for the $j$ party. Every party represented in the POLITICAL IDEOLOGY SPACE is characterized by a mass center depending on the robots which expressed a vote for the party, and the political masses associated to the robots themselves; such a center of mass is obtained using the analogous concept of the classical physics:

$$r_{CM}^{(j)} = \frac{\sum_i m_{i,j} \cdot r_i}{\sum_i m_{i,j}} \tag{12}$$

where the index $i$ describes all the robots of the colony which voted $j$. Figure 5 shows the graphic representation of this situation.

When the scaling process is finished, the coalition which will constitute the new government will be formed by the winning party added by adjacent parties until more of 50% of total votes is reached; the concept of adjacent party is meant in terms of distance among the center of mass of the winning party and the center of mass of the remaining ones. The coalition formation is represented in figure 5 in a hypothetical situation with 3 parties and 11 robots.

### 2.3 Determine Roles

When the coalition is formed, it is necessary to identify what robots are been selected to cover the government roles. In our simulation, we choose the following government roles: *Prime Minister* (PM), *Minister of Defence* (MD), *Minister of Communications* (MC) and *Citizen*. For every government figure, all the robots of the coalition are filtered through the Role's Matrix ($RM$) in order to obtain

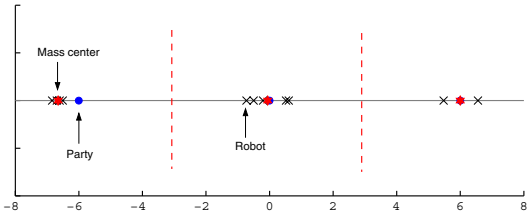



**Fig. 5.** Mass center and coalitions

a sub-matrix made up by the robots of the coalition which are able to execute the tasks concerning such a role. The political roles are assigned on the basis of the following rules: the PM is chosen between the robots belonging to the winning *party*, filtered through the $RM$ matrix. Representing with $r_k^{(PM)}$ the position of the robots which satisfy such a condition, and with $r_{CM}$ the position of the center of mass of the winning party, the PM role is assigned to the robot $i$ closer to the $r_{CM}$ center of mass:

$$R_i = PM \Leftrightarrow r_i^{(PM)} = \min_k |r_k^{(PM)} - r_{CM}| \ . \tag{13}$$

The MD is chosen between the robots belonging to the winning *coalition*, filtered through the $RM$ matrix, which has not assumed a previous governative role; representing with $r_k^{(MD)}$ the position of the robots which satisfy such a condition, the MD role is assigned to the robot $i$ positioned to the right extremity of the coalition:

$$R_i = MD \Leftrightarrow r_i^{(MD)} = \max_k (r_k^{(MD)} - r_{CM}) \ . \tag{14}$$

The MC is chosen between the robots belonging to the winning *coalition*, filtered through the $RM$ matrix, which has not assumed a previous governative role; representing with $r_k^{(MC)}$ the position of the robots which satisfy such a condition, the MC role is assigned to the robot $i$ positioned to the left extremity of the coalition:

$$R_i = MC \Leftrightarrow r_i^{(MC)} = \min_k (r_k^{(MC)} - r_{CM}) \tag{15}$$

The graphic representation of the role's assignment process is showed in figure 6. It is to notice that the robot nearest to the mass center cannot be chosen to cover the PM role because the Roles Matrix does not allow this configuration.

### 2.4 Conduct Business

The robots forming the new government will adopt a behavior in the mission's management to reach the goals in agreement with the political ideologies of the their coalition. The political ideologies are represented by a *strategy* that the robots must adopt. In the MP architecture, we attribute right and left extremes to the two fundamental strategies progressive (typically reactive) and conservative (typically deliberative). In general the government coalition is constituted

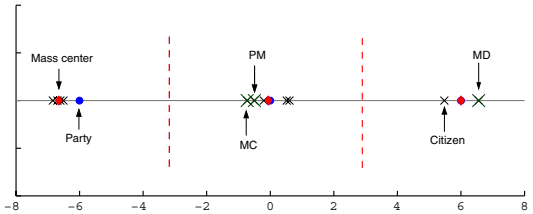



**Fig. 6.** Determine roles

by several parties for which the two strategies are the extremes of a strategy which changes its characteristics depending of the formation of the government. A right-center or left-center government will favor the progressive or conservative strategy basing on the weight given to the right or to the left component in the formation of the government coalition.

A strategy can be characterized by a set of parameters which identify the various aspects of the robot's behavior; for instance we can consider the action ray for a robot's request of mines defusing support, or the maximum time established for the execution of a determinate task, or the margin of discharge of a robot from the subarea assigned in the exploration. Each parameter has values belonging to a continuous interval whose extremes (lower and superior) are associated to the limit left and right strategies. For every parameter $s$, to each of $M$ parties is associated a value so that $s_1 \leq s_2 \leq \ldots \leq s_j \leq \ldots \leq s_M$ where $j$ represents the generic party, while $s_1$, $s_M$ identify the two extreme parties; the winning coalition is constituted from $M'$ parties with $M' \leq M$. The parameter $s_c$ will be only affected by the interested parties, each of which will act on the basis of the weight assumed in the coalition. Its value is calculated as weighted average of the parameters of the coalition parties:

$$s_c = \sum_k a_k \cdot s_k \tag{16}$$

where $k$ refers to the parties which form the coalition. The $a_k$ weight associated to the *k-th* party is obtained taking into account the $V_k$ votes which it received with respect to the total votes of the coalition:

$$a_k = \frac{V_k}{\sum_h V_h} \tag{17}$$

**Mini Crisis.** Mini crisis is a mechanism which allows the partial replacement of the government with new robots belonging to the coalition which won the past elections, holding the same strategy. Such a mechanism allow to avoid the re-election of a new government and wants to solve inefficiencies like the death, damage or excessive loss of energy of any government members, which would negatively affect the behavior of the entire colony. A robot fault involves a change on the Roles Matrix; for instance if the robot *i-th* cannot cover the role *k-th* any

more, then the matrix element $(k, i)$ is updated by a zero value. The mini crisis is generated when a parameter exceeds its limits as well too, for instance when the Welfare (which could give information about the energy of the robot), gets down under a critical threshold which exposes a minimal value to develop its functionalities in a satisfactory way. A mini-crisis is not so critical as a re-election that would punish the coalition, rejecting the strategy played even if the colony behavior was acceptable.

**Re-election.** The re-election mechanism allows the colony to reconfirm the previous coalition or change it totally. The re-election is normally caused by the expiration of the time assigned to the government to complete the entire mission (TIME OUT) or in exceptional case for evident inefficiencies of the robot colony (NO CONFIDENCE), not more imputable to the single robot but to the whole strategy; for instance if a high number of mini-crisis succeeded than there is something wrong in the adopted policy and we need to run a re-election.

## 3 Experimental Results

Unlike robotic architectures which take into account a complex central management of a simple-robot colony, the MP architecture focuses on the complex coordination of robots which present a high autonomy degree. this architecture takes into account for every robot the capability to express a vote in order to elect a party, to form a political coalition, and to identify a strategy played by the govern that allows to accomplish the mission goals.

The MP architecture is based on an FSA established by the following states, realized through MissionLab environment developed at GEORGIA INSTITUTE OF TECHNOLOGY (Georgia Tech) [1, 2]:

1) ELECTION
2) DETERMINE ROLES
3) CONDUCT BUSINESS

The interaction among the states is shown in the figure 7.
The kernel of the architecture is constituted by the ELECTION state which represents a macro-state containing the two following states:

a) VOTING PROCESS
b) COALITION FORMATION

which are responsible respectively for the vote's mechanism for every robot and the formation of the political coalitions which constitute the new governments; the VOTING PROCESS result is related to a set of parameters which characterize each robot; these parameters change during the mission evolution in relation to the quality obtained in the assigned task execution. For each robot can be assigned a low-level task if the robot is a citizen (exploration, bomb defusing) or a high-level task if the robot is a government member (planning, communication). The COALITION FORMATION phase allows to determine the winning

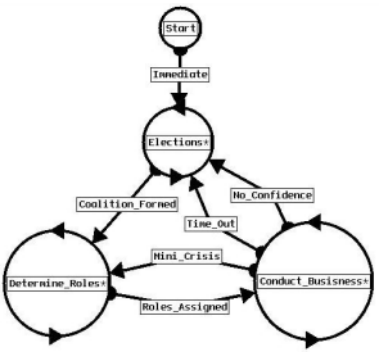


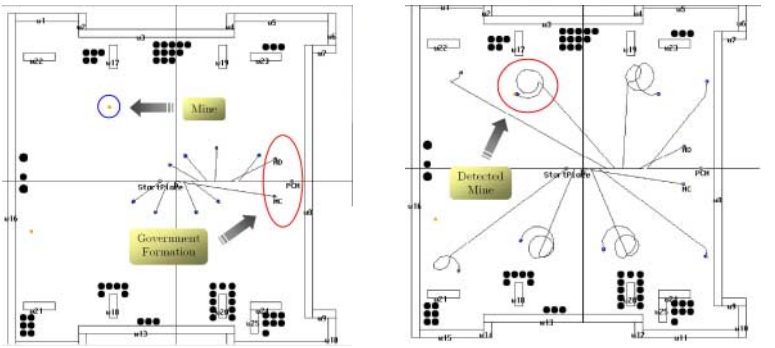

**Fig. 7.** The Robot State Automata Diagram

party on the base of the votes expressed by the single robots; in order to constitute a winning coalition, we select the closest parties by the winning one. The ELECTION state determines the winning political coalition which will constitute the new government. Once the elections have been done and the coalition has been formed, the DETERMINE ROLES state determine which robots will cover the government roles; PM, MD and MC are chosen in order to cover the entire coalition as much as possible thus guarantying the government strength. The CONDUCT BUSINESS state allows the winning government to complete the various task following a strategy which reflects the political trend of the coalition; the government strategy is a compromise between the strategies of the single parties which compose the coalition.

In the FSA of the figure 7 there are triggers which allow to obtain the transitions among the various states during the mission evolution; their meaning is the following:

1. COALITION FORMED: it allows to start the phase of determination of the roles when the elections and the coalition formation are concluded.
2. ROLES ASSIGNED: it allows to carry out the strategy concerning the political trend of the government, once the phase of the roles assignment has been finished.
3. MINI CRISIS: it allows to make a new assignment of the government roles keeping unchanged the political trend of the government.
4. NO CONFIDENCE: it allows to make a new election process caused by a general inefficiency of the whole robot colony.
5. TIME OUT: it allows to make a new election process caused by the expiration of the time assigned to the government to complete the whole mission.

The figure 7 shows the actual design of MP architecture. The main idea is to use an unstructured, dynamic and time-variable environment with movable obstacle. We identify a application such as mines defusing as test-bed application for robotic colony with a high risk for damage to robots. During the testing




(a) Election

(b) Exploration and detect mine

**Fig. 8.** The implemented framework: election, exploration and detect mine

phase of MP architecture we use a scenario composed by 10-20 robots. They do not have prior knowledge about the environments. The robots are divided in two opponents teams. The first team is composed by enemies running the MP architecture. The robots move inside the environment and they put a certain amount of bombs inside the map and after a random time they move the bombs in new positions. The second team, composed of soldiers running the MP architecture, has to find and defuse the bombs placed inside the map from the opponent team, and to detect and kill enemies. The bombs are equipped with a proximity sensor that activate a random timer. If robots will not go to defuse bomb, it will explode and it will kill the robots placed in the neighbors of the bomb. The first phase of our simulation is the phase of coalition formation, where a government for managing mission is selected (see figure 8(a)). Later, the robot citizen performs the typical task of exploration of an unknown environment, where a little part of map (called sub-area) is assigned to each robot. When a robot citizen finds a mine, it must provide defusing too. So it will ask help into an area that will depend to the strategy applied by government (see figure 8(b)). The closest robot to the point of mine discover will so go and help the other robot for mine defusing. During exploration, if a robot finds an enemy, it will try to catch the other in order to make it harmless. A robot consumes an internal resource (*Welfare*) that force itself to spend the minimum energy when it executes a behavior. These features allow us to demonstrate the coordination level of a robot team.

## 4 Summary and Conclusions

One of the fundamental and innovative feature of the Metaphor of Politics Architecture is its dynamic social structure. It is a good balance between the cost

of forming a coalition and the solution quality of a coalition, and also it is capable of forming optimal political coalitions for difficult problem solving under conditions of limited information and resources.

## Acknowledgements

Authors would like to thank Professor Ronald C. Arkin director of Mobile Robot Laboratory from Georgia Tech for the interesting discussions and suggestions about the topics of this research and for the help given during the development of this architecture.

## References

[1] R. C. Arkin, "Cooperation without communication: Multiagent schemabased robot navigation", Journal of Robotic Systems, 9(3), pp. 351-364, (1992). 418

[2] R. C. Arkin and T. Balch, "Aura: Principles and practice in review", Journal of Experimental and Theoretical Artificial Intelligence, 9(2/3), pp. 175-188, (1997). 418

[3] R. C. Arkin, Yoichiro Endo, Brian Lee, Douglas C. MacKenzie, and Eric Martinson "Multistrategy Learning Methods for Multirobot Systems", Mobile Robot, (2003).

[4] Maayan Roth, Douglas Vail, and Manuela Veloso, "A world model for multi-robot teams with communication", under submission, (2003).

[5] M. B. Dias and A. Stentz, "A Free Market Architecture for Distributed Control of Multi-Robot System", 6th International Conference on Intelligent Autonomous Systems (IAS-6), pp. 115-122, (2000).

[6] B. P. Gerkey and M. J. Mataric, "Sold!: Auction methods for multi-robot coordination", IEEE Transaction on Robotics and Automation, special issue on Multi Robot Systems, 18(5), Oct (2002).

[7] S. Carpin and L. E. Parker, "Cooperative Leader Following in a Distributed Multi-Robot System", Proceeding of IEEE International Conference on Robotics and Automation, (2002).

[8] J. Wawerla, G. S. Sukhatme, and M. J. Mataric, "Collective Construction with Multiple Robots", Proceeding of the 2002 IEEE/RSJ International Conference on Intelligent Robot and Systems (IROS'02), Lausanne, Switzerland, September 30 - October 4, (2002).

[9] A. Chella, R. Sorbello, S. Vitabile, A. Lo Grosso, A. Sortino, S. M. Siniscalchi and D. Massara, "A New Paradigm for the Coordination of Bomb Disposal Expert Robot Team", Proceedings of 2002 FIRA (Federation of International Robot Soccer Association), Robot World Congress 26-29 Maggio 2002 Seoul, Corea del Sud, pag. 68-73, (2002). 410

[10] A. Chella, R. Sorbello, S. M. Siniscalchi and S. Vitabile, "MIP: A New Hybrid Multi-Agent Architecture for the Coordination of a Robot Colony Activities", Proceedings of 15th European Conference on Artificial Intelligence (ECAI 2002), 21-26 Luglio 2002, Villerbanne (Lione), Francia, IOS Press Proceeding ed. Frank Van Harmel pp. 713-717, (2002). 410

[11] A. Stoytchev and R. C. Arkin, "Combining Deliberation, Reactivity and Motivation in the Context of a Behavior-Based Robot Architecture", Mobile Robot, (2000).

[12] R. Nair, M. Tambe, S. Marsella, D. V. Pynadath, "Model of Team Formation for Reformation in Multiagent Systems", AAAI Workshop: Coalition Formation in Dynamic Multiagent Environments, Edmonton Alberta Canada, July 29 2002, Proceeding edited by Leen-Kiat Soh, AAAI Press, Menlo Park, California, pp.24-31, (2002).

[13] E. Menegatti, E. Pagello, "Toward a topological mapping with a multi-robot team.", University of Padua, Italy, Proceedings Workshop WS7 of 2002 IEEE/RSJ International Conference on Intelligent Robots and System, (2002).

[14] T. Fong, I. Nourbakhsh, K. Dautenhahn, "A Survey of Socially Interactive Robots", Technical Report CMU-RI-TR-02-29, Carnagie Mellon University, Pittsburgh, PA USA, (2002).

[15] K. Dautenhahn, "Roles and Functions of Robots in Human Society", Implications from Research in Autism Therapy, accepted for publication in ROBOTICA, special issue on Biological Robotics, guest - editor: David McFarland, Cambridge University Press, (2002).

[16] L. Steels "Intelligence - dynamics and representations", In The Biology and Technology of Intelligent Autonomous Agents, Berlin: Sprinter Verlag, (1995).

[17] R. O. Duda, P. E. Hart, D. G. Stark "Pattern Classification", Wiley-Internscience Publication. 412

# Image-Based Monte-Carlo Localisation without a Map

Emanuele Menegatti[1], Mauro Zoccarato[1],
Enrico Pagello[1,⋆], and Hiroshi Ishiguro[2]

[1] Intelligent Autonomous Systems Laboratory
Department of Information Engineering
The University of Padua, Italy
emg@dei.unipd.it
[2] Department od Adaptive Machine Systems
Osaka University
Suita, Osaka, 565-0871 Japan

**Abstract.** In this paper, we propose a way to fuse the image-based localisation approach with the Monte-Carlo localisation approach. The method we propose does not suffer of the major limitation of the two separated methods: the need of a metric map of the environment for the Monte-Carlo localisation and the failure of the image-based approach in environments with spatial periodicity (perceptual aliasing). The approach we developed exploits the properties of the Fourier Transform of the omnidirectional images and uses the similarity between the images to weights the *beliefs* about the robot position. Successful experiments in large indoor environment are presented in which we do not used a priory information on the metrical map of the environment.

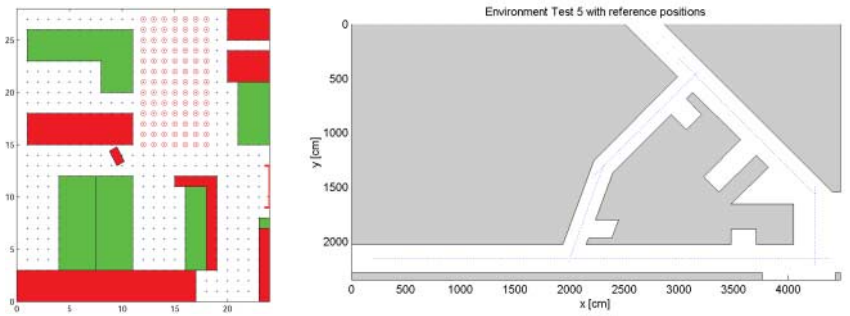
## 1 Introduction

In mobile robotics the localisation problem is a fundamental problem. A robot in order to effectively fulfill an assigned task, must be able to move from point to point and to be able to know its position in the environment. Several solutions have been proposed to this problem. In this work we focused on two successful solutions developed in the past years: the image-based localisation and the Monte-Carlo localisation. Both of these techniques are well known and well known are their limits, i.e. the frame of assumptions under which they are effective. In this work we propose a way to fuse these two techniques trying to overcome, with this fusion, their intrinsic limitations.

The image-based localisation approach [1, 5, 13, 15, 19, 9] is based on the idea that most of the time, the robot does not need a metrical localisation and a topological localisation is enough. We can choose some points in the environment, called **reference locations**. Every time the robot makes a new measurement

⋆ also with Institute of Biomedical Engineering of the National Research Council (ISIB-CNR) Padua , Italy

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 423–435, 2003.
© Springer-Verlag Berlin Heidelberg 2003




(a) (b)

**Fig. 1.** (a) The sketch of the test environment in a laboratory cluttered with many pieces of forniture. The dimensions of the test environment are about $5m \times 2m$. (b) The sketch of the environment where most of the tests were carried out. This is a large environment of about $50m \times 25m$

(i.e. it grabs a new image), it can infer at which of these points it is closer (topological localisation). The robot infers the reference location corresponding to its actual position by comparing the current image it grabbed, called **input image**, with the **reference images** previously grabbed at the reference points. The flaw in this approach is the fact the localisation is purely reactive: to a sensorial input, immediately corresponds a localisation. The image-based localisation approach does not work if more than one reference location corresponds to a input image. In this case the robot cannot choose between the two possible solutions. This happens in environments with highly periodical structures where the appearance of the environment is the same at different locations, like in a maze or in a hospital or in an office building.

The Monte-Carlo localisation approach was developed using range finder sensors like sonars or laser range finders. It is based on the idea of scanning the environment at a certain location and matching this scan to a geometrical map [10]. The possible location of the robot is one of the locations from which the robot would have experienced that scan. The belief about the robot position is managed with a Bayesian approach, as detailed in Section 3. This probabilistic approach allows to manage multi-modal probability distribution of the believed robot position and to fuse the robot's odometry with the range observations. The limit of this approach is that it requires to know a detailed geometrical map of the environment in order to associate a probable robot location to a certain scan.

In this work, we combine the image-based localisation approach with the Monte-Carlo Localisation approach. The advantage is that we do not need a geometrical map of the environment, because the probable position of the robot is calculated using the similarity between the current input image and the reference images. The Monte-Carlo localisation approach allows us to manage the

case in which the current image is similar to more of one reference image (i.e. multi modal probability distribution for the robot position) and to overcome the limitations of the reactive localisation. We will see in Section 3, how the Monte-Carlo localisation keeps track of the history of the robot states by updating the past beliefs about the robot position based on the actual motion performed by the robot.

Other authors proposed a similar fusion. Wolf, Burgard and Burkhardt proposed to use a perspective camera and an image retrieval system for the image-based localisation part. In our opinion, the disadvantage of his method is that it needs a large number of reference images and it needs a geometrical map of the environment [23]. Kröse et al. used an omnidirectional camera and an image matching method based on Principal Component Analysis to reduce memory required to store the reference images. For managing the probability distribution of the robot position, he used a Markov Localisation with an occupancy grid. They focus on the description of the environment rather than on the localisation process. They do not use a model of the motion of the robot and every time the robot takes a new step all positions in the grid have the same probability to contain the robot.

The approach we propose in this paper has been extensively tested in real world indoors environments. We performed preliminary test in the small laboratory of Fig. 1 (a) and more challenging tests in the corridors of Fig. 1 (b). Both are at Floor 7 of Wakayama University, Japan.

## 2 Image-Based Localisation

The main problem tackled by researchers in the image based localisation approach is how to store and to compare a large number of reference images. In fact, if a perspective camera is used, the appearance of the environment from a reference location changes with the direction of gaze. Nevertheless, there are several successful works where perspective cameras have been used. A solution could be to constrain the movements of the robot in order to have the perspective camera gazing always in the same direction [5]. An alternative solution could be to acquire a large number of images with different gaze directions, but to extract from the perspective images compact distinctive features which reduce the amount of memory required to store the images [23].

The best solution in order to reduce the number of reference images is to use omnidirectional cameras. For every reference location we will have only one omnidirectional image, containing the appearance of the environment in all possible gazing directions. The visual memory constructed with the collection of reference images is usually pre-processed to further reduce the memory requirements. The most common approach is to extract a set of eigenimages from the set of reference images and to describe every image as linear combination of these eigenimages. Unfortunately, this approach does not lead to a rotational invariance of the images. In other words, if the image grabbed by the robot is rotated with respect to the corresponding reference image the two images are

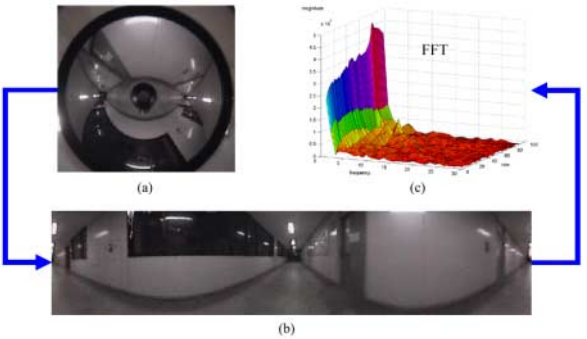



**Fig. 2.** The process of generation of the Fourier Signature. The omnidirectional image (top left) is unwarped in a panoramic cylinder (bottom) and the Fourier coefficient of every row of the panoramic cylinder are calculated (top right)

not longer recognised as similar images. So, authors using this approach need to constrain the heading of the sensor as in [15, 20] or to further preprocess the images in order to obtain the rotational invariance as in [1, 14, 9].

In this work, we decided to reduce the number of images required to describe the environment by mounting an omnidirectional camera on the mobile robot [12, 16, 17]. In order to efficiently store and process the omnidirectional images we exploited the properties of the Fourier transform of panoramic images as detailed in a previous work [13]. Every time an omnidirectional image is grabbed, it is transformed in a compact representation. The calculation of the compact representation is performed as follow (Fig. 2). The omnidirectional image is unwarped creating a **panoramic cylinder**. The panoramic cylinder is expanded row by row into Fourier series. Only the magnitude and the phase Fourier coefficients associated to the lowest frequency components are stored. The result of this procedure is a compact representation of the original omnidirectional image that we called **Fourier signature**. The Fourier signature drastically reduces the amount of memory required to store a view at a reference location and the computational power required to match a current view against the visual memory. In addition, the Fourier signature is the natural representation to implement a rotational invariance, because the magnitude coefficients of the Fourier components are invariants to rotation of the sensor heading. For more details on this procedure, please refer to [13].

We defined a **dissimilarity function** on the Fourier signatures. The higher this value, the more two images are dissimilar. The dissimilarity function is defined as the $L1$ norm of two Fourier signatures:

$$Dis(O_i, O_j) = \sum_{y=0}^{l-1} \sum_{k=0}^{m-1} |F_{iy}(k) - F_{jy}(k)| \tag{1}$$

where $k$ indicates the frequency and $y$ is the index of the row of the panoramic cylinder.

For convenience we defined also a **similarity function** obtained rescaling the dissimilarity values between 0 and 1000.

$$Sim(O_i, O_j) = 1000 - 1000 \frac{Dis(O_i, O_j) - Min_{i,j}\{Dis(O_i, O_j)\}}{Max_{i,j}\{Dis(O_i, O_j)\} - Min_{i,j}\{Dis(O_i, O_j)\}} \quad (2)$$

A robot localisation technique to be used in a populated environment should be able to cope with moving objects, like people, that may cause temporary occlusions in the image. Experiments we conducted, but that we cannot report because of lack of space show the Fourier Signature is robust to occlusion.

As we stated in the introduction, the image-based localisation approach has a flaw: i.e. it fails in environments where two different locations look the same. This is the trick used to get people lost in a labyrinth, i.e. all the junctions of the labyrinth looks the same and one cannot distinguish one junction from another. This situation is called "perceptual aliasing". Nonone of the presented works on image-based localisation can cope with such a situation, except [23]. More information than the simple image similarity is needed to be able to localise the robot. The labyrinth is an extreme example, but also in our everyday-life indoor environment perceptual aliasing is present. In Fig. 3, we present the calculation of the similarity of a current image with respect to all reference images in the environment. The current image is assessed as similar not only to the corresponding image, but is similar also to other reference images at different locations. In order to provide robust image-based localisation, despite the perceptual aliasing we exploited the Monte-Carlo Localisation approach to keep track of the belief about the robot position.

## 3 Monte-Carlo Localisation

To manage the uncertainty about robot position caused by the perceptual aliasing, we adopted a well-known approach based on Monte Carlo Methods. This approach has been successfully used by many researchers [21, 6, 3, 22, 23] and is referred as Monte Carlo Localisation.

This is a probabilistic methods based on Bayesian Filtering (Markov Localisation in robotics) [2, 23, 8]. It calculates the probability density of robot position (the **belief**) and recursively propagates this probability density using motion and perception information. In our implementation motion data come from the odometry sensors and perception data come from the omnidirectional vision system. The belief about robot position is updated every time the robot grabs a new image (i.e. a new observation of the world).

In robotics, many technique has been applied to approximate a continuous density for the beliefs. Two of the most used techniques were Kalman Filter and

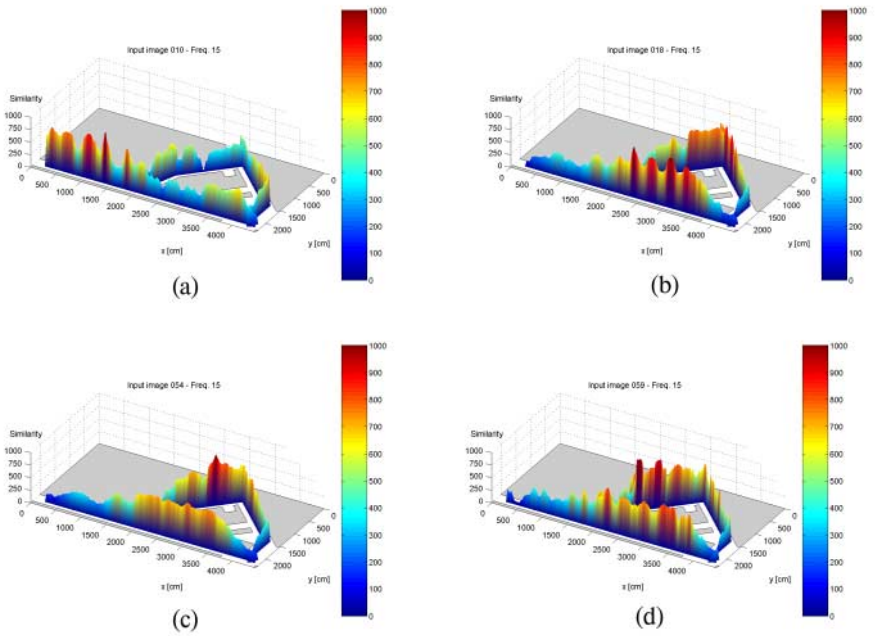



**Fig. 3.** Some examples of perceptual aliasing in our test environment. The colour scale indicate the amount of similarity of the different images: red the heighest similarity, blue the lowest similarity. (a) Input Image 10 matches not only the correct reference image at position 1380 on the x-axis, but also reference images at 320, 620 and 1060. The same happens in plots (b), (c) and (d)

Grid-based Methods, but these methods presented many disadvantages [7] and new techniques have been proposed like the Particle Filter approach.[1]

The Particle Filter approach represents the belief about the robot position with a set of discrete points in the configuration space of the robot. These points are called **samples**. To update the belief over time, these samples are updated. To every sample is associated a weight indicating the probability that the robot is occuping that position. The samples with the lowest weights are deleted and the samples with heighest weights survive and generate new samples.

This approach is efficient and can solve the well know problems of Position tracking and Global Localisation [8]. Nevertheless, Particle Filters alone cannot solve the problem of the Kidnapped robot [8]. In [8] was proposed to continuously generate a subset of samples uniformly distributed in the environment at random positions . This approach relies on the hope that some samples of this subset will be generated by chance close to the position where the robot has been moved.

[1] For a complete review and evaluation of localisation method refer to the two enlightening works of Gutmann [10, 11].

In this case, these samples will have height weights in the next measurements and will survive and cluster the other samples close to them. The drawback of this approach is that it takes several iterations for the samples to cluster around the correct sample and sometimes it takes some iteration to generate a sample close to the position where the robot was kidnapped. In the next section, we will present our solution to speed up the position recovery.

### 3.1 The Proposed Kidnapping Strategy

In order to understand the new kidnapping strategy we propose, we need to briefly review the theory behind the Monte-Carlo Localisation approach. In the Bayesian Filtering problem one has to calculate the probability density of the robot position $Bel(s_t) = p(s_t|O_t, a_t)$ over the time. What is known is: the **prior probability density** $Bel(s_0)$, which describes the initial robot uncertainty about its position; the **prediction model** $p(s_t|s_{t-1}, a_{t-1})$ that applies motion data $a_{t-1}$ to actual state $s_{t-1}$ obtaining a new state $s_t$; the **observation model** $p(O_t|s_t)$ that represents the probability of making observation $O_t$ from state $s_t$.

Using the Bayes Formula and Markov assumption about the state space the equation to calculate the belief is

$$Bel(s_t) = \eta p(O_t|s_t) \int p(s_t|s_{t-1}, a_{t-1}) Bel(s_{t-1}) ds_{t-1} \tag{3}$$

where $\eta$ is a normalisation factor. In Monte-Carlo Localisation the probability density of the robot position (i.e. the belief) is approximated with a set of weighted samples $\{s_t^i, \omega_t^i\}_{i=1\ldots N}$. These samples are updated recursively with a procedure called **sampling-importance-resampling** composed of following steps: For each samples $s_{t-1}^i$ in previous set, a new samples $\hat{s}_t^i$ is drawn from $p(s|s_{t-1}^i, a_{t-1})$ and $s_{t-1}^i$ is replaced with $\hat{s}_t^i$; Given the new observation $O_t$, each sample $\hat{s}_t^i$ is weighted proportionally to $P(O_t|\hat{s}_t^i)$; The samples are replaced proportionally to the weights [4], i.e. if a sample has weight 0.7, the sample is replaced with 7 identical sample.

To tackle the kidnapped robot problem the standard approach is to replace, after the prediction step, a certain amount of samples with others samples randomly drawn in the entire environment. This technique is robust, but in general require many steps to re-localise the robot, expecially in large environments. We propose a new solution: instead of randomly drawing the sample, the new samples should be generated around the reference images that best match the current input image. This assures that the newly generated samples are concentrated around possible locations only. This approach is made possible by the technique we use to match the input image with the reference images. In fact, in the calculation of the similarity, Eq. 2, if we stop the summation over the frequencies at low values of $k$, we obtain a broader localisation, as detailed in [18]. In other words, if we use only the Fourier components corresponding to the lowest spatial frequency, the current input image does not match only the corresponding reference image, but also other similar reference images. Usually,

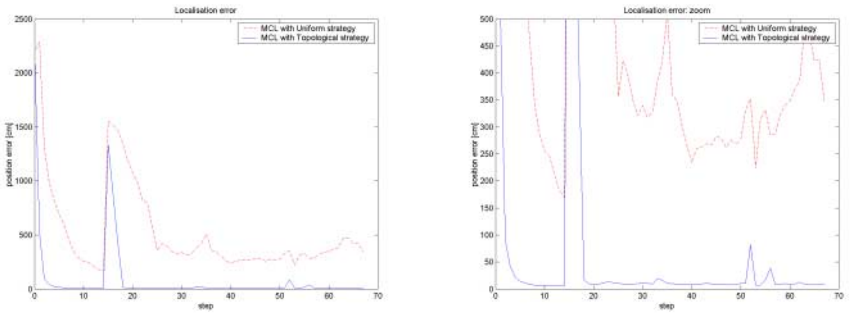



**Fig. 4.** Localisation error during the path of 68 step of Fig. 5(a). The Topological strategy we proposed outperforms the uniform strategy in robustness and speed in recovering the correct position

these reference images are close to the correct reference image, i.e. the new samples concentrate in a region around the correct reference image, but sometimes these similar reference images can be one far away from the other (e.g. in case of perceptual aliasing), i.e. the new samples take into account alternative possibility for the actual location of the robot. In both cases, our system recover very quickly the correct position after the kidnapping. To be noted that our tecnique not only re-localise quickly the robot after a kidnap but also speed-up the global localisation process.

We tested our system in a large indoor environment with a long corridor and a loop (a path about 100 m long): the robot faced many false hypothesis, but the obtained results are good. Particularly we can exploit hierarchical localisation to consider a wide number of hypothesis: that is we can use a lower number of frequencies to generate the new samples with the strategy. Remember that if we use $m$ frequencies in weight process automatically we have all similarity data for $m' <= m$. Using a lower number of frequencies we obtain a broad localisation that is a wide number of hypothesis that after will be controlled by MCL.

## 4 Experiments

We tested our system in the two environment of Fig. 1 (a) and (b). The first environment is a small cluttered laboratory and we use this as a first testbed for our system. We obtained encouraging results in Global Localisation, Position Tracking and Kidnapped Robot problem. We presented these tests in [18]. To confirm the good performances of our system we carried out several experiments in the environment of Fig. 1 (b). This is a very challenging environment. It is a corridor 90 m long with a highly periodical structure and with many locations which look similar, i.e. the same lighting, the same junctions, etc. This means that image-based localisation fails because it could not decide between the various topological localisations as earlier described in Fig. 3. In this environment the

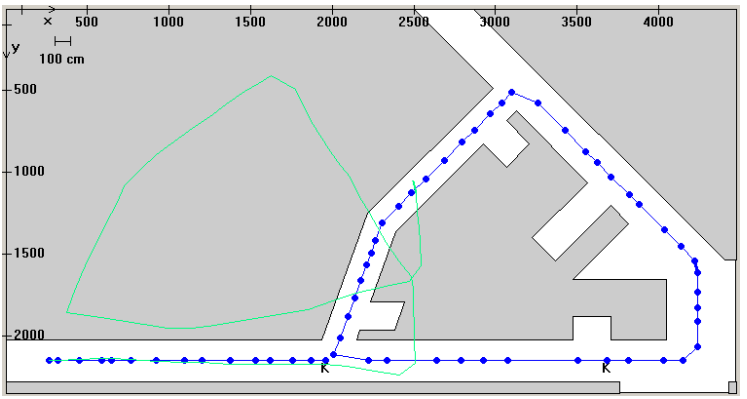



**Fig. 5.** The path we used to test our system in environment of Fig. 1. The blue dotted line is the real path with indicated the positions where the robot grabbed the images. The green line is the path reported by odometry

kidnapping robot problem is particularly challenging, due to this high perceptual aliasing. The **uniform strategy** proposed by Burgard et al. give non-optimal results in this environment (cnf. with [18]). In Fig. 4 can be noted the *uniform strategy* requires many steps to localise the robot from scratch during the Global localisation. In the same way, it requires many steps to re-localise the robot after a kidnapping. This is because, in such a large environmnent, the probability to generate a sample close to the new robot position is low, so it takes many steps to cluster the samples around correct position. This lead to a wide error during the re-localisation process. As it could be seen in Fig. 4 our strategy is more effective. With the mixed technique proposed in this paper, it is possible to speed-up robot global localisation and also to quickly relocalise the robot after the kidnapping. In the experiments of Fig. 7, the robustness of our technique with respect to perceptual aliasing is tested, because the kidnapping is made in a position in the longer corridor with high perceptual aliasing as you could see in Fig. 3 (b). So the re-localisation process have to trial several hypothesis after kidnapping and require 5-6 step to assess the new position.

In Fig. 6 you can see the snapshot of the initial belief at the beginning of the global localisation process. As the system has no information about the robot position, the samples are generated uniformly in the area of the environment. Notice that we don't control if the samples are in an allowed area or not, Fig. 1 (b). This is not necessary. If a sample is inside an object it will be far away from any reference image. Its weight will be zero and at the next step it will be removed by the *sampling-importance-resempling*. In this way we don't use any map of the environment. In Fig. 7 you can see some snapshot of the tests with the uniform strategy (left series of images) and with the proposed topological strategy (right series). As you can see our system localise robot quickly after about 5 steps and after this it continues as a position tracking with clustered samples around

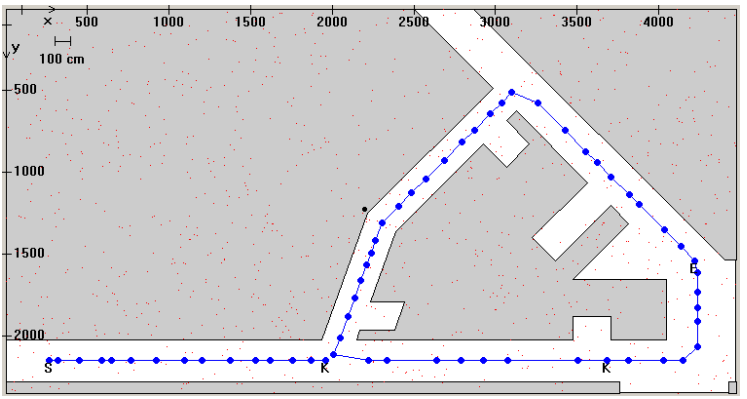



**Fig. 6.** Here you can see the initialisation of the belief in a global localisation task. The samples are generated uniformly in the area without any knowledge of the environment map. Black dot is the current mean estimated position

true position. When the robot is kidnapped our system consider the hypothesis based to vision information and quickly cluster the samples around the true position re-localising the robot in about 4 steps. Instead using uniform strategy it need about 10 steps to re-localise robot. Moreover after re-localisation our system continue as a position tracking whereas with the uniform strategy the samples are generated randomly in the environment and this make less accurate the estimation.

## 5 Conclusions

In this paper we propose a way to fuse the image-based localisation approach based on an omnidirectional sensor with the Monte-Carlo localisation approach. Using an omnidirectional camera we can reduce the amount of memory required to construct a visual memory for the robot because we can reduce the number of reference images with respect to a perspective camera. Using Fourier signature to have a compact representation of an image, we obtain useful properties: a natural rotational invariance, a robustness to occlusion and a hierarchical localisation. To improve our image-based approach in the case of environment with high perceptual aliasing, we fused it with sample-based Monte-Carlo localisation. Particularly we proposed an improvement of this tecnique that exploits the vision information to generate some of the samples. Using hierarchical localisation we can concentrate samples around the true position while considering others hypothesis. We tested this approach in a challenging environment with high perceptual aliasing and we demontrate that our system speed-up the global localisation task, and quickly relocalise after a kidnapping. Moreover, our system does not need to know the geometrical map ogf the environment. We compared our tecnique with the standard uniform approach and showed the advantages

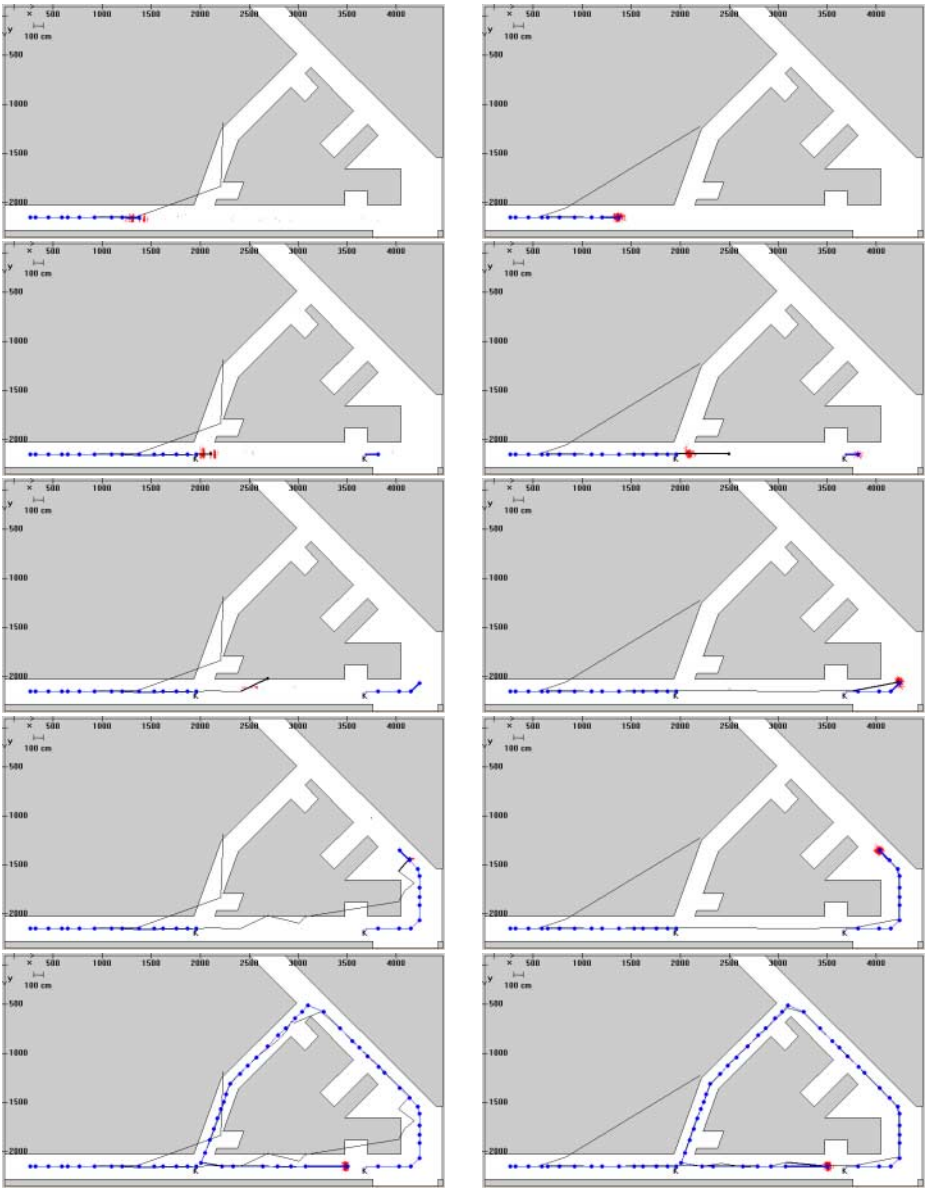



**Fig. 7.** Here you can see some snapshots of our tests. Uniform strategy (left series of images) is compared with our Topological strategy (right series). The blue line is the actual path of the robot and the black curve is the estimated path of the robot. You can see that with our strategy is possible to speed-up robot global localisation and also to quickly relocalise the robot after the kidnapping

of topological strategy. At the moment of writing we are testing our system in a large outdoor environment, particularly to understand the influence of luminosity pattern in our image-based system.

## References


[1] H. Aihara, N. Iwasa, N. Yokoya, and H. Takemura. Memory-based self-localisation using omnidirectional images. In A. K. Jain, S. Venkatesh, and B. C. Lovell, editors, *Proc. of the 14th International Conference on Pattern Recognition*, volume vol. I, pages 1799–1803, 1998. 423, 426

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002. 427

[3] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2000. 427

[4] J. Carpenter, P. Clifford, and P. Fearnhead. An improved particle filter for non-linear problems. *IEEE Proc. Radar, Sonar, Navigation*, vol.146, 1999. 429

[5] R. Cassinis, D. Duina, S. Inelli, and A. Rizzi. Unsupervised matching of visual landmarks for robotic homing unsing fourier-mellin transform. *Robotics and Autonomous Systems*, 40(2-3), August 2002. 423, 425

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo Localization for Mobile Robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999. 427

[7] D. Fox. KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems 14 (NIPS)*. MIT Press, 2001. 428

[8] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of National Conference on Artificial intelligence (AAAI)*, 1999. 427, 428

[9] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transaction on Robotics and Automation*, Vol 16(number 6), December 2000. 423, 426

[10] J.-S. Gutmann, W. Burgard, D. Fox, and K. Konolige. An Experimental Comparison of Localization Methods. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, October 1998. 424, 428

[11] J.-S. Gutmann and D. Fox. An Experimental Comparison of Localization Methods Continued. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), da pubblicare*, 2002. 428

[12] H. Ishiguro. Development of low-cost compact omnidirectional vision sensors. In R. Benosman and S. Kang, editors, *Panoramic Vision*, chapter 3, pages pp. 23–38. Springer, 2001. 426

[13] H. Ishiguro and S. Tsuji. Image-based memory of environment. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-96)*, pages pp. 634–639, 1996. 423, 426

[14] M. Jogan and A. Leonardis. Robust localization using panoramic view-based recognition. In *Proc. of the 15th Int.Conference on Pattern Recognition (ICPR00)*, vol. 4, pages 136–139. IEEE Computer Society, Sept. 2000. 426

[15] B. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appareance-based robot localization. *Image and Vision Computing*, vol. 19(6):pp. 381–391, April 2001. 423, 426

[16] E. Menegatti, F. Nori, E. Pagello, C. Pellizzari, and D. Spagnoli. Designing an omnidirectional vision system for a goalkeeper robot. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: Robot Soccer World Cup V.*, pages pp. 78–87. Springer, 2002. 426

[17] E. Menegatti and E. Pagello. Omnidirectional mirror design based on the vision task. In *Proceeding of 1st Nat. Conf. "Sistemi Autonomi Intelligenti e Robotica Avanzata*, pp.151–159, Frascati - Roma, Oct. 2002. 426

[18] E. Menegatti, M. Zoccarato, E. Pagello, and H. Ishiguro. Hierarchical image-based localisation for mobile robots with monte-carlo localisation. In *Proc. of European Conf. on Mobile Robots (ECMR'03)*, (to appear), Sept. 2003. 429, 430, 431

[19] T. Pajdla and V. Hlaváč. Zero phase representation of panoramic images for image based localization. In F. Solina and A. Leonardis, editors, *8-th International Conference on Computer Analysis of Images and Patterns*, number 1689 in Lecture Notes in Computer Scinence, pages 550–557, Tržaška 25, Ljubljana, Slovenia, September 1999. Springer Verlag. 423

[20] A. Rizzi and R. Cassinis. A robot self-localization system based on omnidirectional color images. *Robotics and Autonomous Systems*, 34: 2001. 426

[21] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. In *Intern. Journal of Robotics Research*, Vol. 19, pp. 972–999, Nov. 2000. 427

[22] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo Localization for Mobile Robots. *Artificial Intelligence Journal*, 128(1-2), 2001. 427

[23] J. Wolf, W. Burgard, and H. Burkhardt. Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *Proc. of the IEEE Intern. Conf. on Robotics & Automation (ICRA)*, 2002. 425, 427

# Making Explicit the Hidden Semantics of Hierarchical Classifications

Bernardo Magnini, Luciano Serafini, and Manuela Speranza

ITC-irst Istituto per la Ricerca Scientifica e Tecnologica
Via Sommarive 18 - Povo
38050 Trento, Italy
{magnini,serafini,manspera}@itc.it

**Abstract.** Hierarchical classifications are concept hierarchies used to organize large amounts of documents. File systems, products' taxonomies for the market place and the directories provided by Web portals are common examples of hierarchical classifications. We propose a methodology for building a semantic interpretation of hierarchical classifications on the basis of the analysis of the taxonomic relations and the linguistic material they contain. We provide a formal semantics for hierarchical classifications and use it to interpret the implicit knowledge represented. Relevant phenomena addressed include the disambiguation of polysemous words, the semantics of multiwords, and the interpretation of coordinations. We report about experiments performed on the Web Directories of Google and Yahoo!.

## 1 Introduction

Hierarchical classifications are taxonomic structures used to organize large amounts of documents. The most typical examples of hierarchical classifications are file systems, market place catalogs, and the directories of Web portals. Documents of a hierarchical classification can be of many different types, depending on the characteristics and uses of the hierarchy itself. In file systems, documents can be any kind of file (e.g. text files, images, applications, etc); in the directories of Web portals, documents are pointers to Web pages; in the market place, catalogs organize either product cards or service titles.

Hierarchical classifications are quite useful for document classification and retrieval. Users browse hierarchies of concepts and quickly access the documents associated with the different concepts. The content of a concept is typically described by a label, but it also depends on the concepts at higher levels in the hierarchy, even though the relations between concepts are usually not explictly labeled.

Hierarchical classifications are now widespread as knowledge repositories and the problem of their integration and interoperability is acquiring a high relevance from a scientific and commercial perspective. A typical application of hierarchical classification interoperability occurs when a set of companies want to exchange products without sharing a common product catalog. In these cases the best

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 436–448, 2003.
© Springer-Verlag Berlin Heidelberg 2003

solution is to find mappings between their catalogs [10]. In [4], we have proposed an algorithm that finds semantic relations between the nodes of different hierarchical classifications. This algorithm strongly relies on a linguistic analysis of the labels contained in the classifications. The main difference between this algorithm and other approaches to schema matching such as [1], [6] and [3] is that in order to interpret a node of a hierarchy we do not limit ourselves to a linguistic analysis of its labels. Instead, we extend this analysis by considering the *implicit information* deriving from the *context* where the node occurs, i.e. the structural relations with the other nodes of the hierarchy.

Indeed, one of the most evident peculiarities of hierarchical classifications (and in general of schemas) is that the meaning of a node depends not only on the label of the node, but also on the position of the node in the hierarchy. Like databases and ontologies, concept hierarchies are built on taxonomic relations between concepts, but such relations are implicit and have to be interpreted. Like plain texts, hierarchical classifications contain linguistic material, i.e. labels that can be analyzed with NLP techniques; the context provided for a label, however, is not a sentence or a paragraph, but a set of concepts placed at different levels. Consequently, the interpretation is performed in two steps: first, each single concept is analyzed separately from the others and is associated with a basic logical form. Then, on the basis of its position in the hierarchy and of its relations with other nodes, we build a full logical form for each concept.

For instance, we can have a hierarchical classification of documents about sports that contains a node labeled with *Sports Organizations*, and a second hierarchical classification on the same topic, containing a node labeled with *Organizations* which is a child of a node labeled with *Sport*. Clearly the semantics of the node "Sports Organizations" in the first hierarchy coincides with the semantics of the node "Organizations" in the second hierarchy . This equality however cannot be discovered by simply analyzing the labels. One has to discover that the arc connecting "Sports" and "Organizations" is a specification arc. This interpretation is very ambiguous and context dependent. Consider the example of a node labeled with *Schools* with a descendant node labeled *United States*, in this case the hierarchical relation between the two nodes has to be interpreted as a location relation. This interpretation is based on the semantics that is hidden in the labels and in the hierarchical structure.

The aim of this paper is to describe a method to analyze the implicit knowledge hidden in hierarchical classifications and to make it explicit in order to provide a correct interpretation of its concepts. In particular, we describe an algorithm that, given a concept hierarchy, returns an interpretation of each node of a hierarchy in terms of a logical formula of a description logics [2]. The ideal output of our algorithm for the concept hierarchy of Figure 1 is reported in Figure 2. This algorithm does not consider the documents classified under the nodes, so that it can be used in situations where such information is partially available or not available at all.

The paper is structured as follows. In Section 2 we provide a formal definition of concept hierarchy based on intuitions on how documents are classified by

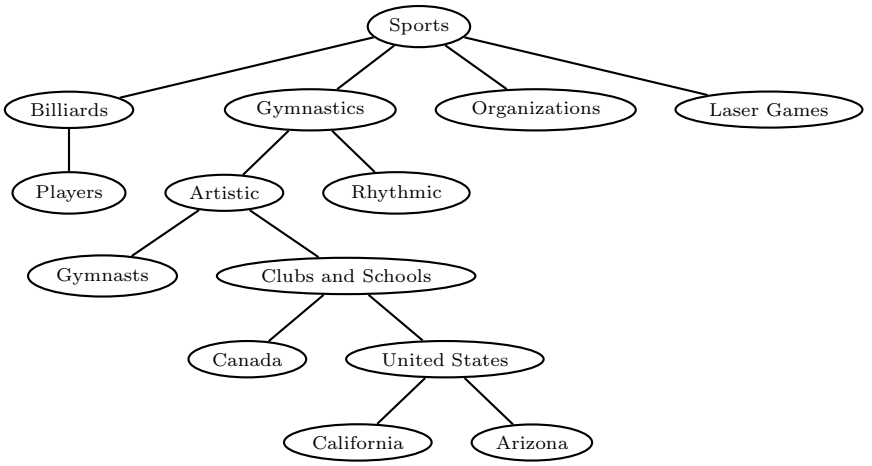



**Fig. 1.** Example of concept hierarchy (from Google Web Directories)

humans. In Section 3 we describe the analysis of the concepts performed without considering the hierarchical structure of the context. In Section 4 we describe the interpretation of the concepts based on the structural relations of the concept hierarchy. In Section 5 we discuss the results of an evaluation experiment where the procedure is applied to the Web Directories of Yahoo! and Google. Finally, Section 6 reports some relevant related work.

## 2 Concept Hierarchies

Here we introduce more formally the terms of our problem (see [11] for a more detailed description).

**Definition 1 (Concept hierarchy)** *A* concept hierarchy *is a triple* $H = \langle C, E, l \rangle$ *where* $C$ *is a finite set of nodes,* $E$ *is a set of arcs on* $C$*, such that* $\langle C, E \rangle$ *is a rooted tree, and* $l$ *is a function from* $C$ *to a set* $L$ *of labels expressed in natural language.*

An example of concept hierarchy is provided in Figure 1, where a small part of the category 'Sport' in the Web Directories of Google is represented.

**Definition 2 (Hierarchical classification)** *A* hierarchical classification *of a set of documents* $\Delta$ *in a concept hierarchy* $H = \langle C, E, l \rangle$ *is a function* $\mu : C \rightarrow 2^{\Delta}$.

Classifications guide users in retrieving documents from the whole set $\Delta$. The common procedure for seeking documents in a hierarchical classification is by

entering the hierarchy from the root node, and, at each node, by choosing the child node under which the document is more likely to be classified. This choice is based on a semantic interpretation of the labels associated with the nodes, so that in most cases users do not need to check the content of the documents.

Consider, for instance, the concept hierarchy of Figure 1. In order to find documents about Romanian artistic gymnasts, a user would start from the root labeled with *Sports*, would first select *Gymnastics*, then *Artistic*, and then *Gymnasts*, and would finally retrieve the documents classified under this node. Users' choices are guided by the following facts:

- Understanding of the meaning of the labels attached to the nodes encountered during the navigation; in this case, *Sports, Billiards, Gymnastics, Organizations, Artistic, Rhythmic, Gymnasts* and *Clubs and Schools*.
- Knowledge of the fact that Romanian gymnasts are artistic gymnasts, and that gymnastics is a sport.
- Assumption that a document about artistic gymnasts is much more likely to be classified under the sub-tree rooted at *Sports/Gymnastics* than under the ones rooted at *Sports/Billiards* and *Sports/Organizations*. Similar assumptions are related to the choice between the children of *Sports/Gymnastics*, and so on.
- Awareness that the node *Gymnastics* is the most specific node about the topic 'Romanian gymnasts', as there is no node *Romania*.

In order to be useful for a user, a classification should therefore respect a number of classification criteria, which can be summarized as follows:

M1 Each concept $c \in C$ has a *meaning* $m(c)$, which is some entity of a world domain.
M2 The meaning of a concept $c$ depends only on the labels associated with a finite set of nodes $F(c) \subset C$ called the *focus of* $c$.
C1 A document $\delta \in \Delta$ is classified under a descendant node $c'$ of $c$, i.e., $\delta \in \mu(c)$, only if $\delta$ is concerned with $m(c')$.
C2 A document $\delta$ is classified under the node $c$ if $c$ does not have any descendant $c'$ such that $\delta$ is concerned with $m(c')$.

Criterion M1 guarantees that the hierarchical classification is done with a domain model in mind which is assumed to be shared by the users. Criterion M2 guarantees that the meaning of the concepts can be determined by visiting a finite (and possibly small) subset of the whole classification. Criteria C1 and C2 are standard classification criteria which can be found for instance in Yahoo! or in the Open Directory Project. These two criteria provide the connection between the meaning of the labels and the set of documents classified under the node.

To formalize criteria M1-2 and C1-2, we provide a notion of *meaning* of a concept in a real world. In order to express meanings we adopt a logical approach, i.e. meanings are expressed in terms of formulas of a description logic $\mathcal{DL}$, so that a label of a node can be associated either with a concept expression,

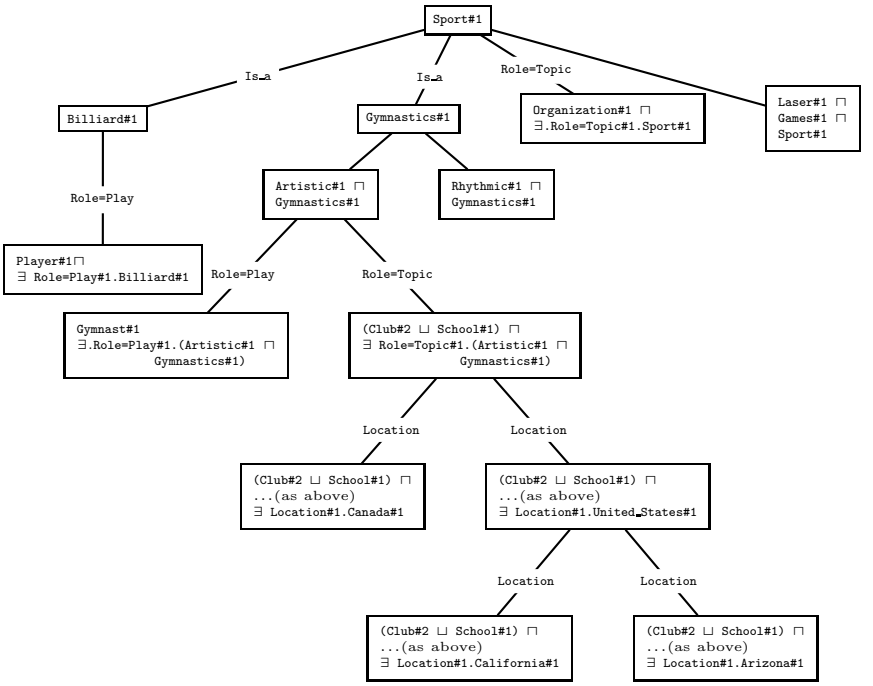



**Fig. 2.** The concept hierarchy of Figure 1 enriched with the meanings, and the relation types

or with a role description, or with an individual constant of $\mathcal{DL}$. The $\mathcal{DL}$ we adopt in this work is the description logics containing $\{\top, \bot, \sqcap, \sqcup, \neg, \exists R.$ and $\forall R.\}$. Furthermore, in the current work a concept $c$ is associated with a meaning $m(c) = \phi$ which is a concept (roles and constants are not yet considered). For instance, the node *Sports* in Figure 1 corresponds to the primitive concept `sport`, while the node labeled with *Organizations* is associated with the concept $\texttt{Organization} \sqcap \exists \texttt{role} = \texttt{topic.sport}$, as it refers to sports organizations whose core business is sport.

This analysis is performed by using the semantic information provided in WordNet [7], which has been adopted because it is the largest repository of word senses and semantic relations currently available. The primitive concepts and roles of $\mathcal{DL}$ are selected among WordNet senses, on the basis of the labels occurring in $f(c)$. For instance, for the label *Sports*, WordNet provides 5 senses, which represent 5 different concepts of $\mathcal{DL}$. Again, for the word 'topic', WordNet provides different senses; for the role, we chose `topic#1`, meaning 'subject, theme'; another example of role is `location#1`, which describes spatial relations between concepts. The process of finding the WordNet senses composing the label interpretation for a concept $c$ is based on a linguistic analysis

of $l(c)$. In order to minimize the ambiguity a further filtering of such senses is performed.

We build a basic label interpretation for the single concepts (see Section 3), and then we construct the contextual interpretation of every concept $c \in H$ by combining the basic interpretation of the concepts and their ancestors (see Section 4). The best we can obtain through the contextual interpretation of a concept hierarchy is an interpretation of each node such that the two criteria C1 and C2 are satisfied. For instance, the ideal output for the concept hierarchy in Figure 1 is reported in Figure 2. This paper proposes a first step towards a complete procedure for knowledge discovery in hierarchical classifications. In particular, we have addressed the construction of the logical formulas for concepts, but we haven't addressed the discovery of implicit roles. For instance, in the current version of the algorithm, the node labeled with *Organizations* is associated with the concept $[\texttt{sport\#1}] \sqcap [\texttt{organization\#1}]$.

## 3 Basic Label Interpretation

Concepts in concept hierarchies are described by labels, which in turn are composed by words and, possibly, separators between them. Labels are taken from a wide variety of linguistic expressions. Labels can be single words, noun phrases, such as *Local Currency Systems*, prepositional phrases, such as *Sociology of Religion*, verb phrases, etc. More complex labels can also contain conjunctions, punctuation, and acronyms.

In the first phase of our algorithm, we parse the labels attached to the nodes and generate a formula in $\mathcal{DL}$ representing a first approximation of the meaning of the node.

**Definition 3 (Basic label interpretation)** *The* basic interpretation *is a function $lm : L \rightarrow \Phi_{\mathcal{DL}}$, that maps each single label $l \in L$ in a concept description $lm(l)$ of the description logic $\mathcal{DL}$ whose concepts are taken from the set of* WordNet *senses.*

Intuitively, a basic interpretation provides an interpretation of the concept label as a stand alone object. For instance, the basic interpretation of the node labeled with *Organizations*, occurring under *Sports*, would be equal to the basic interpretation of a node *Organizations* occurring under *Billiards*.

The basic interpretation of the labels is performed in two phases. The first step of the procedure consists of text chunking, i.e. dividing each label into syntactically correlated parts of words. For this we run the Alembic chunker [5], developed by MITRE Corporation as part of the Alembic extraction system. For example, with the label *Science Fiction and Horror*, the chunker first selects a part of speech for each word ('Science', 'Fiction', and 'Horror' are nouns, 'and' is a conjunction); then, it identifies two noun groups (NGs), i.e. 'Science Fiction' and 'Horror' (the syntactic head is marked in small capitals), and a coordinating conjunction between them (1a).

(1) *a.* $[(Science)_{nn}(\text{FICTION})_{nn}]_{NG}(and)_{cc}[(\text{HORROR})_{nn}]_{NG}$

The output of the chunker is used to transform each label into a basic logical form. A noun group consisting of more than one word is interpreted as the conjunction of the head and all its modifiers. For instance, $[(Science)_{jj}(\text{FICTION})_{nn}]$ is interpreted as [`Science` $\sqcap$ `Fiction`], the reason being that the documents classified under a node with such label should be concerned both with 'science' and with 'fiction'.

The relations between different noun groups are interpreted on the basis of the linguistic material connecting them: coordinating conjunctions and commas are interpreted as a disjunction; prepositions, like 'in' or 'of', are interpreted as a conjunction; expressions denoting exclusion, like 'except' or 'but not', are interpreted as negations.

For example, *Science Fiction and Horror* is interpreted as a disjunction (2a), since under that node there might be both documents about 'science fiction' and documents about 'horror'; on the other hand, *Professional Photographers of America* and *Garments except Skirts* are examples of conjunction and negation respectively.

(2) *a.* [`Science` $\sqcap$ `Fiction`] $\sqcup$ [`Horror`]

In order to perform the semantic interpretation of the labels we access WORDNET [7]. When a word is found, all the senses of that word are selected and attached to the basic logical form. In the case of *Science Fiction and Horror*, for instance, WORDNET provides the senses of all the three nouns, and we obtain the logical form in (3a).

(3) *a.* [`science`$*$ $\sqcap$ `fiction`$*$] $\sqcup$ [`horror`$*$]

In our notation, `trade*` denotes the disjunction `trade#1` $\sqcup$ `trade#2` $\sqcup$ ... $\sqcup$ `trade#n` of all the senses of 'trade' in WORDNET; `trade#3` indicates sense 3 of 'trade'.

**Multiwords.** When two or more words in a label are contained in WORDNET as a single expression (i.e. a multiword), the corresponding senses are selected and, in the basic logical form, the intersection between the two words is substituted by the multiword. For instance, 'Science Fiction' is provided in WORDNET as a single expression, so the logic interpretation is substituted by the senses of the multiword (4a).

(4) *a.* [`science_fiction`$*$] $\sqcup$ [`horror`$*$]

**Word Sense Disambiguation.** Since multiwords are much less polysemous than single words, the recognition of the multiwords provided in WORDNET is a first step towards word sense disambiguation. A second step is performed by

exploiting the relations between senses provided in WordNet. The label *maple tree*, for instance, is first transformed into [`maple*` $\sqcap$ `tree*`], since 'maple' and 'tree' are polysemuous words and 'maple tree' is not provided in WordNet as a multiword. However, `maple#2` (defined as 'any tree or shrub of the genus Acer') is a second level hyponym of `tree#1` (i.e. 'tree' as a woody plant), and so the meaning of 'tree' as a diagram (i.e. `tree#2`) can be discarded.

## 4 Contextual Interpretation

An interpretation of a concept in a hierarchical classification as a stand alone object, however, is partial, as the meaning of a node depends on the context where the node occurs, i.e. its focus (see criterion M2). Intuitively, the focus $f(c, H)$ of a node $c$ belonging to $H$ is the part of $H$ that the user is required to visit in order to understand whether a document is in $c$. In practice, we have defined the focus of $c$ as the set of its ancestors with their direct descendants. For instance, the focus of the node *Artistic* in Figure 1, includes *Gymnastics*, *Rhythmic*, *Sports*, *Billiards*, *Organizations* and *Laser Games*. The contextual interpretation of a node $c$ gives a meaning to the node on the basis of the meaning of the nodes belonging to its focus.

Let $\mathcal{H}$ be the class of concept hierarchies, and $\mathcal{C}$ the class of nodes occurring in some $\mathcal{H}$.

**Definition 4 (Contextual interpretation)** *A* contextual interpretation *is a function* $m : \mathcal{C} \times \mathcal{H} \rightarrow \Phi_{\mathcal{DL}}$*, where* $\Phi_{\mathcal{DL}}$ *is a concept of a description logics* $\mathcal{DL}$*, such that* $f(c, H) = f(c', H')$ *implies that* $m(c, H) = m(c', H')$*.*

Once we have associated a contextual meaning $m(c)$ to a concept $c$, we can define the class of documents classified under $c$ to be the set $\Delta(c)$ such that for each $\delta \in \Delta(c)$ containing the document $\delta$ satisfying the following conditions:

C1: One of the main topics of $\delta$ is a concept $\phi$, and $\phi$ subsumes $m(c)$, i.e., $\phi \sqsubseteq m(c)$;
C2: For any descendant $c'$ of $c$, $\phi$ does not subsume $m(c')$, i.e., $\phi \not\sqsubseteq m(c)$.

Given the concept hierarchy $H$, the main task described in this Section is to find a proper contextual interpretation $m(c, f(c))$ by combining the linguistic analysis of the labels associated with $c$ and $f(c)$, with the information provided by the structure of $f(c)$.

The full logical form of a node $c$ is the conjunction of the basic logical forms of $c$ and all its ancestors. As an example, let's take a concept hierarchy with the root *Soccer*, a descendant *Leagues*, and a further descendant *Clubs*. The full logical form of the root is simply [**soccer***], while the full logical forms of *Leagues* and *Clubs* contain conjunctions (5a and 5b respectively).

(5) *a.* [`soccer`$*$] $\sqcap$ [**league**$*$]
*b.* [`soccer`$*$] $\sqcap$ [`league`$*$] $\sqcap$ [**club**$*$]

**Implicit Disjunctions.** However, things can become more complicated when dealing with conjunctions and negations. As explained before, the presence of a coordinating conjunction makes the disjunction between noun groups explicit, but we can also have implicit disjunction between elements placed at different levels of the hierarchy (concepts with a disjoint descendant).

In the previous example, for instance, there are two conflicting interpretations: from the point of view of the hierarchical structure, *clubs* denotes a subset of *leagues* (being a child of it); on the other hand, from the point of view of the world knowledge provided in WordNet, [`club#2`] and [`league#1`], can be considered as disjoint because they have the same hypernym, i.e. `association#1`. In order to combine the two information sources, *Leagues* has to be reinterpreted as if it were *Leagues and Clubs* (6a), while the interpretation of *Clubs* (5b) is not modified.

$$(6)\ \ a.\ [\texttt{soccer}*] \sqcap [[\textbf{league\#1}] \sqcup [\textbf{club\#2}]]$$

When two concepts in a path are disjoint, i.e. when a concept is disjoint from a concept that is either an ancestor or a descendant, the meaning of the ancestor has to be reinterpreted. More formally:

Let $c$ and $c'$ be two concepts, and let `c#i` and `c'#j` be two senses of $c$ and $c'$ respectively. We apply the following rule: replace the sense `c#i` with `c#i` *and* `c'#j`, if `c'#j` is disjoint from `c#i` and $c$ is an ancestor of $c'$.[1]

**Implicit Negations.** Similarly, negations can be marked by expressions like 'but not' or 'except', but can also be implicit in the case of elements belonging to different labels (inclusion relation berween two siblings). For instance, in Google Web Directories we have *Sociology* and *Science* as sibling nodes classified under *Academic Study of Soccer*; from the point of view of world knowledge, sociology is a science (and in fact in WordNet `sociology#1` is a second level hyponym of `science#2`). As a consequence, the node labeled with *Science* has to be interpreted as if it were *Science except Sociology.*

Whenever a concept in a label has a part-of relation or an is-a relation with a concept in another label on the same level, it is necessary to re-interpret the meaning of the more general concept. More formally:

Let $c$ and $c'$ be two concepts, and let `c#i` and `c'#j` be two senses of $c$ and $c'$ respectively. We apply the following rule: replace the sense `c#i` with `c#i` $-$ `c'#j`, if `c#i` is either a hyponym (i.e. it is a more specific concept) or a meronym (i.e. it denotes a part) of `c'#j` and $c$ and $c'$ are siblings.

**Multiwords in Context.** The recognition of multiwords can also be performed on different contiguous levels. For instance, in WordNet there is a multiword 'billiard player', so in a hierarchy where *Sports* has *Billiards* as a child and

[1] `s#k` is disjoint from `t#h` if `s#k` belongs to the set of opposite meanings of `t#h` (if `s#k` and `t#h` are adjectives) or, in the case of nouns, if `s#k` and `t#h` are different hyponyms of the same synset.

*Players* as a further descendant, in the full logical form representing *Players* (7a), the conjunction of [`billiard*`] and [`player*`] can be substituted with the multiword (7b).

(7) *a.* [`sport`$*$] $\sqcap$ [`billiard`$*$] $\sqcap$ [**player**$*$]
*b.* [`sport`$*$] $\sqcap$ [`billiard`_**player**$*$]

**Word Sense Disambiguation in Context.** We perform word sense disambiguation by taking into consideration both structural relations between labels and conceptual relations between words belonging to different labels.

Let L be a generic label and $L^1$ be either an ancestor label or a descendant label of L and let `s*` and $\mathtt{s}^1$`*` be respectively the sets of WordNet senses of a word in L and a word in $L^1$. The algorithm checks whether one of the senses belonging to `s*` is either a synonym, a hyponym, a meronym, a hypernym or a holonym of one of the senses belonging to $\mathtt{s}^1$`*`; if so, the two senses are stored in a table. This search is repeated for all words in the hierarchy. Then, if one or more senses belonging to a set are found in the table, those senses are retained and all other senses are deleted from the set; otherwise, if no sense is found, no filtering can be performed on that set.

As an example, imagine *Apple* (which can denote either a fruit or a tree) and *Food* as its ancestor; since there exists a hyponymy relation between `apple#1` (denoting a fruit) and `food#1`, we retain `apple#1`.

## 5 Evaluation

As a test set for the evaluation of the algorithm for the interpretation of concept hierarchies, which has been implemented in Java, we have focused on the Web Directories of Yahoo! and Google, where documents are represented by millions of Web page URLs'.

Yahoo! and Google Web Directories have respectively fourteen and fifteen main categories (e.g. 'Society & Culture' and 'Science' in Yahoo! and 'Arts', 'Health' and 'News' in Google) consisting of a number of nodes ranging between a few thousand and tens of thousands nodes each. Each of these categories can be considered as the root of a sub-hierarchy.

A preliminary analysis has been performed on two sub-hierarchies, i.e. 'Architecture' (under the main category 'Art') and 'Medicine' (under the main category 'Health'), whose sizes range between one hundred and one thousand nodes (cfr. Table 1). The labels attached to the nodes are generally short, with an average of 1.5-1.8 words per label. Labels can be repeated more than once since the same label can be attached to different nodes in different places of the hierarchy; in fact, the bigger a hierarchy is, the higher is the average repetition of the labels.

These two sub-hierarchies have been chosen among those where WordNet's coverage was highest, i.e. between 88.7% and 95.5%. It has been found that that

**Table 1.** Analysis of the Architecture and Medicine sub-directories in Yahoo! and Google

| | Yahoo! Arch. | Google Arch. | Yahoo! Med. | Google Med. |
|---|---|---|---|---|
| # Concepts | 105 | 312 | 703 | 1,023 |
| Average label repetition | 1.0 | 1.3 | 2 | 1.8 |
| # Words | 170 | 521 | 1,231 | 1,549 |
| # Words/label | 1.6 | 1.7 | 1.8 | 1.5 |
| WordNet's coverage | 95.5% | 91.5% | 88.7% | 91.4% |
| Average polysemy | 3.8 | 3.7 | 4.6 | 3.2 |
| # Multiwords | 11 | 45 | 51 | 116 |
| # Disjunctions | 10 | 51 | 99 | 58 |
| # Conjunctions | 41 | 109 | 239 | 325 |

each lemma has on average between 3.2 and 4.6 senses, which makes the need for word sense disambiguation very important.

A manual and partial evaluation of the disambiguation performed by the system has shown that the procedure is very precise, with a precision rate ranging between 69 and 75%, but has a low recall, which is due to the fact that WordNet contains only hyponymy and meronymy relations and no other kinds of relations, like role or location relations. In the specific case of Yahoo! 'Architecture', average polysemy has been reduced from 3.8 to 2.9 senses/word after word sense disambiguation.

As for the presence of multiwords, between 11.3% and 14.4% of the labels contain a multiword, which is remarlable, if we consider that between 54.8% and 62.6% of them consist of one single word. The good recognition of multiwords contributes to reduce the polysemy of concepts as most of them (almost 80% of the multiwords recognized in the experiment) are monosemous.

As far as negations are concerned, the hierarchies under analysis do not contain expressions denoting exclusion, while many implicit negations have been discovered. Our procedure works quite well with nouns denoting concrete objects like buildings. For example, in Google 'Architecture', the node *Architecture/History/Periods and styles/Romanesque* has two descendants, *Churches* and *Cathedrals*; since cathedrals are actually churches (and an hyponymy relation between them is provided in WordNet), *Churches* is reinterpreted as if it were *Churches except Cathedrals*. On the other hand, performance is not so good with abstract nouns.

A limitation of the current system is that the use of the Alembic chunker does not permit the resolution of coordination ambiguities involving nominal compounds (and neither would the use of a more sophisticated parser, as this problem has received relatively little attention [9], when compared to other aspects, like for instance prepositional phrase attachment). To make an example, noun phrase coordinations with the form *n1 and n2 n3* admit two structural

analysis, one in which *n1* and *n3* are the two syntactic heads being conjoined (8a) and one in which the conjunction is between the modifiers *n1* and *n2* (8b).

$$(8)\ \begin{array}{ll} a & (Nightclubs)\,and\,(Dance\,Halls) \\ b & (Food\,and\,Drug)\,Administration \end{array}$$

Most of the times Alembic suggests the first analysis, which is correct in the case of *Nightclubs and Dance Halls* (9a), but is incorrect in many other cases, like *Food and Drug Administration* (9b) , which should be analyzed as in 9c. We plan to refine our analysis of coordinations involving nominal compounds by introducing rules based on number agreement as in [9].

$$(9)\ \begin{array}{ll} a & [(\textsc{Nightclubs})_{nn}]_{NG}(and)_{cc}[(Dance_{nn}(\textsc{Halls})_{nn}]_{NG} \\ b & *[(\textsc{Food})_{nn}]_{NG}(and)_{cc}[(Drug)_{nn}(\textsc{Administration})_{nn}]_{NG} \\ c & [((Food)_{nn}(and)_{cc}(Drug)_{nn})(\textsc{Administration})_{nn}]_{NG} \end{array}$$

## 6 Related Work

Our use of linguistic analysis to enrich hierarchical classifications with semantic information is comparable to the work presented in [12] on conceptual indexing. In this work the conceptual structure of phrases is analyzed using semantic relationships between words to establish connections between the terms in a conceptual taxonomy. Even if similar methodologies are applied, our approach aims at interpreting already existing taxonomies in order to make explicit a number of semantic relations, while in conceptual indexing the starting point is the extraction of terminology from documents.

The problem of allowing the interoperability of concept hierarchies, and in particular of catalogs, has been addressed also in [1]. Their approach is based on the use document classification algorithms; our methodology, on the other hand, does not use the documents associated with the nodes of the conceptual hierarchy, since it is based on the interpretation of labels and of the relations between them.

Finally, research related to the linguistic analysis of multi-word expressions and terminology has been conducted by Jacquemin and Morin in [8], who describe a framework for organizing multi-word candidate terms with the help of automatically acquired links between single-word terms derived from WordNet.

## 7 Conclusions

We have provided a formal semantics for hierarchical classifications and then used that formal framework to explore a number of linguistic issues crucial for interpreting the knowledge implicitly represented in such classifications.

The methodology we have proposed, based on a linguistic interpretation of the labels provided in the hierarchy, takes as input a concept hierarchy and

returns the interpretation of each label. The process of interpreting a label coincides with the progressive construction of a logical form in description logics, where predicates are WordNet senses. It is performed in two steps: on the basis of the output of the chunker, basic logic forms are first built for each single concept independently of the others; then, full logical forms are built by combining the basic logic form of each concept with the basic logical forms of the nodes belonging to its focus.

In the future we plan to work on a systematic analysis of the performance of the methods with respect to the different steps and on the realization of a module for the discovery of the different kinds of relations between concepts, such as role, location, etc.

## References

[1] Agrawal, R. and Srikant, R.: On Integrating Catalogs. Proc. of the Tenth International World Wide Web Conference (WWW-2001), Hong Kong, China, May, 2001

[2] Baader, F. and Nutt, W.: Description Logic Handbook. Pages 47-100, Cambridge University Press.

[3] Bergamaschi, S., Guerra, F. and Vincini, M.: Product Classification Integration for E-Commerce. Proc. of WEBH-2002, Second International Workshop on Electronic Business Hubs. Aix En Provence, France. September, 2002

[4] Bouquet, P., Magnini, B., Serafini, L. and Zanobini, S.: A SAT–based Algorithm for Context Matching. To appear in: Proc. of the Fourth International and Interdisciplinary Conference on Modeling and Using Context

[5] Day, D. S. and Vilain, M. B.: Phrase Parsing with Rule Sequence Processors: an Application to the Shared CoNLL Task. Proc. of CoNLL-2000 and LLL-2000. Lisbon, Portugal, September, 2000

[6] Doan, A., Madhavan, J., Domingos, P. and Halevy, A.: Learning to Map between Ontologies on the Semantic Web. Proc. of WWW-2002, 11th International World Wide Web Conference, Honolulu, Hawaii, May, 2002

[7] Fellbaum, C. (ed.): WordNet: An Electronic Lexical Database. The MIT Press, Cambridge, US, 1998

[8] Jacquemin, E. and Morin, E.: Projecting Corpus-Based Semantic Links on a Thesaurus. Proc. of the 37th Annual Meeting of the Association for Computational Linguistics, University of Maryland, June, 1999

[9] Resnik, P.: Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. Journal of Artificial Intelligence Research 11, 2001

[10] Schulten, E., Akkermans, H., Botquin, G., Dörr, M., Guarino, N., Lopes, N. and Sadeh, N.: Call for Participants: The E-Commerce Product Classification Challenge. IEEE Intelligent Systems, 16-4, 2001

[11] Serafini, L., Bouquet, P. and Donà, A.: Ctxml Context Markup Language. Technical report, ITC-irst, 2002

[12] Woods, W. A.: Conceptual Indexing: A Better Way to Organize Knowledge. SUN Technical Report TR-97-61, 1997

# Sub-symbolic Encoding of Words

Giorgio Vassallo[1], Giovanni Pilato[2], Andrea Maggio[3], Alessandro Puglisi[3], and Salvatore Gaglio[2,3]

[1] Centro di Ricerche Elettroniche in Sicilia
Via della Regione Siciliana - Monreale (PA) - Italy
vassallo@cres.it
[2] Istituto di CAlcolo e Reti ad alte prestazioni
Italian National Research Council
Viale delle Scienze - 90128 Palermo - Italy
g.pilato@icar.cnr.it
[3] Dipartimento di ingegneria INFOrmatica
University of Palermo
Viale delle Scienze - 90128 Palermo - Italy
gaglio@unipa.it, {maggio,puglisi}@csai.unipa.it

**Abstract.** A new methodology for sub-symbolic semantic encoding of words is presented. The methodology uses the WordNet lexical database and an *ad hoc* modified Sammon algorithm to associate a vector to each word in a semantic $n$-space. All words have been grouped according to the WordNet lexicographers' files classification criteria: these groups have been called lexical sets. The word vector is composed by two parts: the first one, takes into account the belonging of the word to one of these lexical sets; the second one is related to the meaning of the word and it is responsible for distinguishing the word among the other ones of the same lexical set. The application of the proposed technique over all the words of WordNet would lead to an interesting instrument for the sub-symbolic processing of texts. The first experimental results show the effectiveness of the proposed approach.

## 1 Introduction

A very interesting area, in the field of text mining, is statistical natural language processing[1, 2, 3]; in particular, it would be useful to design a technique for sub-symbolic processing of texts[4, 5, 6, 7]: the numeric encoding of words is the most natural approach.

The simplest method to associate uncorrelated codes to words is to assign a unit vector for each token.[8] However this method is not manageable when a large number of words has to be considered. To tackle this problem, Honkela et al. [8] introduced a technique for encoding words in such a manner that the $i$-th word in a sequence of words was represented by a 270-dimensional real vector with random number components. The SOM algorithm has been used for creating word category maps describing relations of words based on their

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 449–461, 2003.
© Springer-Verlag Berlin Heidelberg 2003

contexts. The result of the processing showed that interrelated words which have similar contexts appear close to each other on the map.

In recent years a framework has been developed, called Latent Semantic Analysis (LSA)[9], to extract and represent the meaning of words by statistical computations applied to a large corpus of texts. Text is represented as a matrix where rows are associated to words, while columns are associated to documents or other context. The content of the $(i,j)$-th cell is a function of the $i$-th word frequency in the $j$-th document or other context; then the technique of Singular Value Decomposition is applied to the matrix of word-context co-occurrence statistics for mapping words in an high dimensional space.[9]

Another framework, similar to LSA, for developing high dimensional vector representations of words based on a co-occurrence analysis of large samples of written texts, is the Hyperspace Analogue to Language[10] method. A different methodology, called *Random Indexing* has been developed by[11] for constructing context vectors representing the distributional profiles of words. Other approaches can be found in[12, 13, 14, 15].

In this paper a new methodology for sub-symbolic semantic encoding of words is introduced: the peculiarity of this approach is the use of a lexical database instead of plain text as source of information for encoding a word as a vector in a semantic space.

The proposed technique uses the WordNet lexical database[16] and the Sammon algorithm[17] properly modified to associate a sequence of $n$ real numbers representing a vector in a $n$-space with each word, so that semantically near words will be also close points in this $n$-dimensional space.

All the words have been grouped according to the WordNet lexicographers' files classification criteria: these groups are here called *lexical sets*. The vector in the $n$-space is composed by two parts: the first one, called *lexical part*, takes into account the belonging of the word to one of the lexical sets; the second one, called *semantic part*, is related to the meaning of the word and it is responsible for distinguishing the word among the others of the same lexical set.

The lexical part is built in a prearranged way. The semantic part is calculated using the *ad hoc* modified Sammon algorithm. To calculate the semantic part, it is necessary to introduce also a semantic similarity measure between the words. To do this, a set of terms, called *descriptive set*, describing each word has been associated to each of them. The descriptive set is determined by the information retrieved from the WordNet semantic net. The Tanimoto formula[19], which is a similarity measure between sets, has been then used in the Sammon algorithm to calculate the similarity between the words.

In order to validate the effectiveness of the proposed approach, it has been analysed the syntactic category of nouns, obtaining satisfactory results: in particular the analysis of the "*noun.motive*" lexical set is here illustrated. The application of the proposed technique to all of the words in WordNet would lead to an interesting instrument for the sub-symbolic processing of texts.

This paper is organized in the following way: in the next section the theoretical background about WordNet and the Sammon projection algorithm are

reported; in section 3 it is presented the proposed technique; finally, in section 4 experimental results are illustrated.

## 2 Theoretical Background

### 2.1 WordNet

WordNet is a lexical database whose design is inspired by the current psycholinguistic theories of human lexical memory.[16] WordNet is not merely a thesaurus, because it organizes lexical information in terms of word meanings in a semantic net.

WordNet organizes nouns, verbs, adjectives and adverbs into synonymous sets, which are further arranged into a set of "*lexicographers' files*" by syntactic category and other organizational criteria (i.e. nouns indicating actions, artefacts, objects, nouns related to the animal world, verbs of cognition, communication, emotion, etc) as reported in the table 1.

The term "*word*" is usually used for referring both to the orthographic expression and its associated meaning. WordNet reduces such ambiguity using the term "*word form*" for referring to the physical utterance or inscription and "*word meaning*" to the associated lexicalised concept. A word meaning can be represented by simply listing the word forms that can be used to express it: this set of synonyms is called "*synset*". In order to specify the concept associated to a synset, a short gloss is usually connected to the synset itself. It is possible to discern the different meanings of a word form by the "*sense number*". The sense number is an integer that WordNet associates to each word form to distinguish its different meanings.

In this paper, the term "*word*" indicates the word form associated with its sense number. Therefore, one word form with different sense numbers represents different "*words*".

**Table 1.** WordNet Lexicographers' files list

| $N^\circ$ | Lexicographers' file | $N^\circ$ | Lexicographers' file | $N^\circ$ | Lexicographers' file |
|---|---|---|---|---|---|
| 00 | adj.all | 15 | noun.location | 30 | verb.change |
| 01 | adj.pert | 16 | noun.motive | 31 | verb.cognition |
| 02 | adv.all | 17 | noun.object | 32 | verb.communication |
| 03 | noun.Tops | 18 | noun.person | 33 | verb.competition |
| 04 | noun.act | 19 | noun.phenomenon | 34 | verb.consumption |
| 05 | noun.animal | 20 | noun.plant | 35 | verb.contact |
| 06 | noun.artifact | 21 | noun.possession | 36 | verb.creation |
| 07 | noun.attribute | 22 | noun.process | 37 | verb.emotion |
| 08 | noun.body | 23 | noun.quantity | 38 | verb.motion |
| 09 | noun.cognition | 24 | noun.relation | 39 | verb.perception |
| 10 | noun.communication | 25 | noun.shape | 40 | verb.possession |
| 11 | noun.event | 26 | noun.state | 41 | verb.social |
| 12 | noun.feeling | 27 | noun.substance | 42 | verb.stative |
| 13 | noun.food | 28 | noun.time | 43 | verb.weather |
| 14 | noun.group | 29 | verb.body | 44 | adj.ppl |

$N^\circ$ = Lexicographers' file number

### 2.2 Sammon Algorithm

The algorithm of Sammon[17] is a non linear mapping technique (NLM) of $N$ vectors from a high-dimensional $L$-space to a lower-dimensional $l$-space so that the inherent data structure is approximately preserved. The dimensionality of the destination space is usually $l = 2$ or 3 so that resultant data configuration can be evaluated easily by human observations. The input of the algorithm is a list of $N$ $L$-vectors. Then the algorithm calculates a distance matrix $\mathbf{D}$ and initialises $N$ $l$-vectors, each of them corresponds to a specific vector in the $l$-space. Such initialisation is often formed by the $l$ components of the input vectors with largest variances; otherwise a random initialisation can be used.

The NLM algorithm changes the $l$-space configuration using a steepest descent procedure to search for a minimum of the Sammon distortion error:

$$E = \frac{1}{\sum_{i<j}^{N}(d_{ij}^*)} \sum_{i<j}^{N} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*} \tag{1}$$

where $d_{ij}^*$ is the distance between the $i$-th vector and the $j$-th vector of the initial $L$-space, and $d_{ij}$ is the distance between the $i$-th vector and the $j$-th vector of the destination $l$-space.

## 3 The Proposed Solution

The idea is to encode each word as a sequence of $n$ numbers representing a vector in a $n$-space, so that semantically near words will be also close points in this $n$-dimensional space that will therefore constitute a semantic space.

The underlying hypothesis of the proposed procedure is that the words belonging to a synset will be treated as one same word, since they represent the same meaning. Therefore, the words belonging to the same synset will be represented by the same number, called here "*synset number*".

The proposed solution consists of building a $n$-vector, called "*word vector*", representing a word as composed by two parts: the first one, called "*lexical part*", indicating the lexical information, while the second one, called "*semantic part*", indicating the semantic information.

The first part is constituted by $n_1$ elements, the second one by $n_2$ components (see formula (2) ). Therefore a word will be represented as a point in a space of dimensionality $n = n_1 + n_2$. The whole process is illustrated in fig. 1 and will be explained in detail in the following sub-paragraphs.

$$\text{word vector} = \underbrace{(\overbrace{x_0, x_1, \cdots, x_{n_1-1}}^{\text{Lexical part}}, \overbrace{y_0, y_1, \cdots, y_{n_2-1}}^{\text{Semantic part}})}_{n\text{-vector}} \tag{2}$$

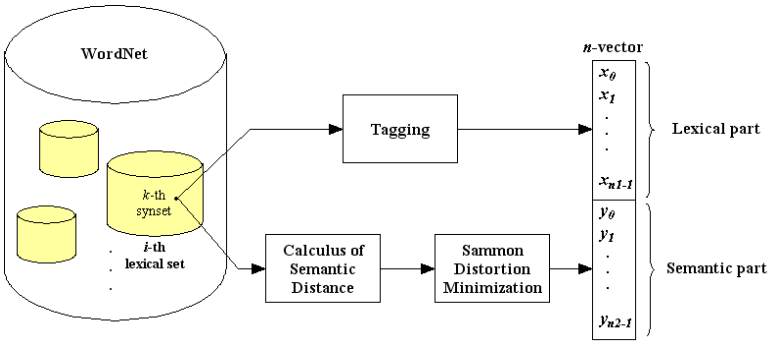



**Fig. 1.** Procedure of encoding the $k$-th word of the $i$-th lexical set

### 3.1 The Lexical Part

All words have been grouped according to the WordNet lexicographers' files classification criteria (see table 1) in order to point out their lexical category: these groups of words are therefore called here "*lexical sets*".

The lexical part of the word vector is a binary vector and it concerns the belonging of a word to one of these sets of words.

**Tagging.** The dimension $n_1$ of the lexical part corresponds to the 45 lexical sets (see table 1); therefore, if a word belongs to the $i$-th lexical set, the corresponding lexical part will be tagged by a 1 in the $i$-th position and by 0 in the rest of the lexical part, i.e.:

$$x_i = \begin{cases} 1 & \textit{if the word belongs to the i-th syntactic set} \\ 0 & \textit{otherwise} \end{cases} \tag{3}$$

### 3.2 The Semantic Part

To each word of WordNet it has been associated a set of other word forms describing it: these sets of word forms are here called "*descriptive sets*".

A descriptive set is determined using the WordNet semantic net and it is composed by:

- the word forms of the same synset of the word;
- the nouns and the adjectives in the gloss of the synset (removing the example sentences);
- the word forms of the hypernym synset.

Because several uninflected words can be found in the gloss, each noun and adjective of the gloss has been transformed in its base form using the JWNL (Java WordNet Library) morphological processor[18].

The semantic part of the word vector is based on these descriptive sets and it is related to the meaning of the words. While the lexical part is responsible to distinguish the words among different lexical sets, the semantic part is responsible to distinguish the words belonging to the same lexical set.

Let $S_i(k)$ be the descriptive set of the $k$-th word belonging to the $i$-th lexical set, let $M_i$ be the number of words belonging to the $i$-th lexical set, and let $s_i(k)$ be a vector of dimensionality $n_2$ corresponding to the semantic part of the $k$-th word of the $i$-th lexical set.

To calculate the semantic part, it is necessary:

1. to introduce a semantic distance measure between words exploiting the Tanimoto formula[19];
2. to minimize the Sammon distortion formula using the introduced semantic distance between words.

**Calculus of the Semantic Distance.** To express the semantic distance between words, it has been used the Tanimoto measure[19] that is a measure of similarity between sets. The semantic distance between the words $k$ and $j$ belonging to the $i$-th lexical set, is defined as:

$$\delta_i\left(k,j\right) \equiv 1 - \frac{\left|S_i\left(k\right) \cap S_i\left(j\right)\right|}{\left|S_i\left(k\right) \cup S_i\left(j\right)\right|} \tag{4}$$

Sometimes the descriptive sets of two words with different meaning occasionally present some words in common. This leads to a value of $\delta_i\left(k,j\right)$ smaller than 1. For this reason the distance $\delta_i\left(k,j\right)$ has been modified so that:

$$\delta_i^\theta\left(k,j\right) = \begin{cases} \delta_i\left(k,j\right) & \textit{if } \delta_i\left(k,j\right) \leq \theta \\ 1 & \textit{otherwise} \end{cases} \tag{5}$$

where the value of the constant $\theta$ is experimentally determined.

It can be defined a matrix $\Delta_i^\theta$ , associated with the $i$-th lexical set, so that its generic element is the semantic distance between the words $k$ and $j$:

$$\Delta_i^\theta \equiv \left[\delta_i^\theta\left(k,j\right)\right] \tag{6}$$

where $k$ and $j$ are ranging from 1 to $M_i$, therefore $dim\left(\Delta_i^\theta\right) = M_i$. The matrix $\Delta_i^\theta$ is square, symmetric and with a null diagonal.

**Sammon Distortion Minimization.** After computing the matrix $\Delta_i^\theta$, the $M_i$ vectors $s_i(k)$ have to be calculated. To reach this goal it has been properly modified the traditional Sammon algorithm[17]; in particular:

1. the dimensionality $l$ of the destination space has been increased to $n_2$ (with $n_2 > 3$) and it is experimentally determined;
2. input data is a distance matrix $\mathbf{D}_i^*$ whose generic element $[k,j]$ is calculated as it follows[3]:

$$d_i^*\left(k,j\right) \equiv e^{\alpha \cdot \delta_i^\theta\left(k,j\right)} \tag{7}$$

where $\alpha$ is an experimentally determined constant greater than zero, and $\delta_i^\theta(k,j)$ is given by the formula (5). In this way, the range of distance values has been expanded from $[0,1]$ to $[1,e^\alpha]$.

After a random initialisation of the $M_i$ vectors $s_i(k)$, the Sammon distortion minimization technique is applied.

## 4 Experimental Results

In the proposed solution a word vector, formed by a lexical part and by a semantic part, is assigned to each word.

The calculus of the lexical part is very simple and therefore it does not need to be analysed furthermore, since it is built in a prearranged way and it has only the purpose of distinguishing the words among the lexical sets.

Instead, it is useful to analyse the semantic part, obtained by the modified Sammon algorithm, verifying that, in the $n_2$ dimensional space, near vectors correspond to synonymous words, whereas distant vectors correspond to words with different meanings.

Experimental results highlight that, for the generation of the semantic part, a good choice is $\theta = 0.9$ ( formula (5) ), $n_2 = 10$ ( formula (2) ) and $\alpha = 5$ (formula (7)).

In order to validate the effectiveness of the proposed approach, the vectors $s_i(k)$ produced by the *ad hoc* modified Sammon algorithm and representing the words in the $n_2$ dimensional space have been projected in a bi-dimensional space using the traditional version of the Sammon algorithm. The Sammon algorithm has been selected because it is simple, efficient and it allows to analyse easily the obtained results[17].

The mapping in the bi-dimensional space introduces some distortions, that have to be considered when the points, representing the words, are analysed. This test is useful to verify if near points correspond to synonyms and if far points correspond to words with a different meaning.

The first results, obtained with the syntactic category of nouns, are satisfactory. In the following paragraph the analysis of the *"noun.motive"* lexical set is illustrated.

### 4.1 Analysis of the *"noun.motive"* Lexical Set

The words of the lexical set number 16 (see table 1) belong to the syntactic category of nouns and they point to *"the reason for the action or that which gives purpose and direction to behaviour"* according to the WordNet lexical database[16].

It is possible to gather the synsets so that all the synsets of a group have a relationship of direct hypernymy with one synset, called *"representative synset"*. The representative synset does not necessarily belong to the represented synset group.

In table 2 are illustrated the synset groups of the *"noun.motive"* lexical set, indicating also the word forms associated with their sense numbers between

**Table 2.** Synset groups of "*noun.motive*"

| Gr | Nr | Word form *(sense number)* |
|---|---|---|
| *A* | **17** | **mania(1), passion(4), cacoethes(1)** |
| | 18 | agromania(1) |
| | 19 | dipsomania(1), alcoholism(2), potomania(1) |
| | 20 | egomania(1) |
| | 21 | kleptomania(1) |
| | 22 | logorrhea(1), logomania(1) |
| | 23 | monomania(1), possession(4) |
| | 24 | necrophilia(1),necrophilism(1) necromania(1) |
| | 25 | phaneromania(1) |
| | 26 | pyromania(1) |
| | 27 | trichotillomania(1) |
| *B* | **10** | **irrational motive(1)** |
| | 15 | irrational impulse(1) |
| | 17 | mania(1), passion(4), cacoethes(1) |
| | 29 | compulsion(2), obsession(1) |
| *C* | **11** | **urge(1), impulse(1)** |
| | 12 | abience(1) |
| | 13 | adience(1) |
| | 14 | death instinct(1), death wish(1) |
| | 28 | wanderlust(1), itchy feet(1) |
| *D* | **33** | **conscience(1),scruples(1), moral sense(1), sense of right and wrong(1)** |
| | 34 | wee small voice(1), small voice(1), voice of conscience(1) |
| | 35 | sense of shame(1), sense of duty(1) |
| *E* | **31** | **ethical motive(1),ethics(1) morals(1), morality(2)** |
| | 32 | hedonism(1) |
| | 33 | conscience(1),scruples(1), moral sense(1), sense of right and wrong(1) |
| | 36 | inner light(1),light(13), light within(1),christ within(1) |
| *F* | 1 | life(12) |
| | 2 | rational motive(1) |
| | 10 | irrational motive(1) |
| | 11 | urge(1), impulse(1) |
| | 31 | ethical motive(1), ethics(1), morals(1), morality(2) |
| | 37 | psychic energy(1), mental energy(1) |
| *G* | **2** | **rational motive(1)** |
| | 3 | reason(1), ground(2) |
| | 7 | incentive(1), inducement(1), motivator(1) |
| | 9 | disincentive(1), deterrence(1) |
| *H* | **7** | **incentive(1),inducement(1) motivator(1)** |
| | 8 | moral force(1), dynamic(1) |
| *I* | **3** | **reason(1), ground(2)** |
| | 4 | occasion(3) |
| | 5 | score(5), account(8) |
| | 6 | why(1), wherefore(1) |
| *J* | **29** | **compulsion(2),obsession(1)** |
| | 30 | onomatomania(1) |
| *K* | **38** | **incitement(3), incitation(1) provocation(2)** |
| | 39 | signal(2) |
| *L* | **37** | **psychic energy(1) mental energy(1)** |
| | 38 | incitement(3), incitation(1) provocation(2) |
| | 40 | libidinal energy(1) |
| *M* | **40** | **libidinal energy(1)** |
| | 41 | cathexis(1), charge(14) |
| | 42 | acathexis(1) |
| *N* | **15** | **irrational impulse(1)** |
| | 16 | compulsion(1), irresistible impulse(1) |

Gr = group name
Nr = synset number

round brackets. Column *"Gr"* indicates the group names, column *"Nr"* indicates the synset numbers, whereas the representative synset is indicated in bold text.

The representative synset of the group **F** does not belong to the lexical set under process, therefore it has not been shown.

To point out the meaning of each group, the WordNet definitions of the representative synsets are reported:

- **group *A***:"*an irrational but irresistible motive for a belief or action*";
- **group *B***:"*a motivation that is inconsistent with reason or logic*";
- **group *C***:"*an instinctive motive*";
- **group *D***:"*a motivation deriving logically from ethical or moral principles that govern a person's thoughts and actions*";
- **group *E***:"*a motivation based on ideas of right and wrong*";

- **group** ***F***:"*the psychological feature that arouses an organism to action toward a desired goal*";
- **group** ***G***:"*a motive that can be defended by reasoning or logical argument*";
- **group** ***H***:"*a positive motivational influence*";
- **group** ***I***:"*a rational motive for a belief or action*";
- **group** ***J***:"*an irrational motive for performing trivial or repetitive actions against your will*";
- **group** ***K***:"*something that incites or provokes*";
- **group** ***L***:"*an actuating force or factor*";
- **group** ***M***:"*psychic energy produced by the libido*";
- **group** ***N***:"*a strong spontaneous and irrational motivation*".

To better explain the procedure for calculating the semantic part, let us consider two synsets:

- *Nr'= 17 ( mania(1), passion(4), cacoethes(1) )*;
- *Nr" = 21 ( kleptomania(1) ).*

The queries of the two synsets in the WordNet database give the following results:

**mania, passion, cacoethes** – (*an irrational but irresistible motive for a belief or action*)
=> irrational motive – (*a motivation that is inconsistent with reason or logic*)

**kleptomania** – (*an irresistible impulse to steal in the absence of any economic motive*)
=> mania, passion, cacoethes – (*an irrational but irresistible motive for a belief or action*)

The words in boldface before the symbol "–" are the word forms of the synsets 17 and 21, the sentence between parenthesis is the gloss, whereas the symbol "=>" points to the direct hypernym synset. This information allows to create the descriptive sets of the two synsets:

$S_{16}(17)$ = *(mania, passion, cacoethes, irrational, irresistible, motive, belief, action, irrational motive)*

$S_{16}(21)$ = *(kleptomania, irresistible, impulse, steal, absence, economic, motive, mania, passion, cacoethes)*

The subscript number 16 is the lexical set number of "*noun.motive*" (see Table 1).

To calculate the similarity between the two synsets, the proposed methodology proceeds as it follows:

1. the number of elements belonging to the intersection of the two sets $S_{16}(17)$ and $S_{16}(21)$ is counted, i.e. the number of word forms that the two sets have in common and then it is counted the number of word forms belonging to the union between the two sets, then the semantic distance is calculated according to the formula (4):

$$\delta_{16}(17,21) = 1 - \frac{|S_{16}(17) \cap S_{16}(21)|}{|S_{16}(17) \cup S_{16}(21)|} = 1 - \frac{5}{14}$$

2. the $\delta_{16}^{\theta}(17,21)$ value is derived from the formula (5) and is used in the formula (7) to determine the distance between the above-mentioned synsets:

$$d_{16}^{*}(17,21) \equiv e^{\alpha \cdot \delta_{16}^{\theta}(17,21)}$$

After the calculus of the distance matrix $\mathbf{D}_i^*$ , it is applied the steepest descent technique[17] to minimize the Sammon distortion error in a $n_2 = 10$ dimensional space. The obtained results will constitute the sub-symbolic encoding of the semantic part of the words belonging to the "*noun.motive*" lexical set.

To test the validity of the illustrated approach the traditional Sammon projection algorithm has been used on the semantic parts generated by the proposed methodology. The results are reported in the graph shown in fig. 2. In this graph the synset groups of table 2 have been pointed out. To each group it has been associated the corresponding WordNet definition of the representative synset. By this graph it can be noted that the relationships of direct hypernymy among synsets are often respected, however there are some positioning errors; in particular:

**a.** the synsets number 15 and 16 are too distant, therefore the group $N$ has not been drawn;
**b.** the synset number 37 is too distant by the other ones belonging to the group $L$ therefore the synset 37 has not been inserted into the group L;
**c.** the synset number 3 is too distant by the other ones belonging to the group $G$ therefore the synset 3 has not been inserted into the group G;
**d.** the group $J$ is too lengthened.

Of course these inaccuracies, in the graph, could be due to distortions introduced by the traditional Sammon algorithm. The obtained results are however satisfactory, because words that hava similar meanings are grouped together.

## 5 Conclusions and Future Work

In this paper it has been introduced an innovative methodology for sub-symbolic semantic encoding of words. The proposed technique uses the WordNet lexical database[16] and the *ad hoc* modified Sammon algorithm[17] to associate to each word a sequence of $n$ real numbers representing a vector in a $n$-space, so that semantically near words will be also close points in this $n$-dimensional space.

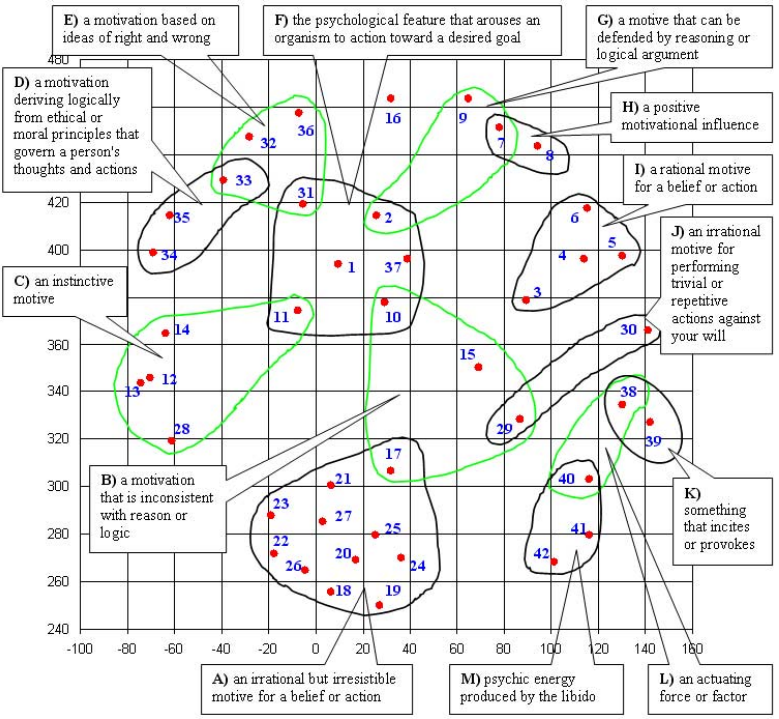



**Fig. 2.** Synsets of "*noun.motive*" grouped according to the relationships of direct hypernymy

The proposed approach differs from the SOM-Based[8], LSA[9], HAL[10], and other solutions[11, 12, 13, 14, 15]. It is used a standard lexical database manually built instead of a document corpus, and each word is encoded as the union of two parts: the first one related to the lexical category of words and the second one, of relatively low dimensionality, related to the meaning of words. Finally, each word form can have more than one associated vector.

The next goal is to extend the encoding to other syntactic categories (verbs, adjectives and adverbs). The application of the proposed technique to all WordNet words would lead to an interesting instrument for sub-symbolic processing of texts, including automatic classification, indexing, organization and research in unorganised text repositories.

## References

[1] Bellegarda, J. R.: Exploiting latent semantic information in statistical language modelling. Proceedings of the IEEE , Volume: 88 Issue: 8, Aug 2000; pp. 1279 -1296. 449

[2] Hofmann, T.: Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization. Advances in Neural Information Processing Systems, S. A. Solla, T. K. Leen and K. R. Muller (eds), 2000, pp.914-920, MIT press 449

[3] Pilato, G., Sorbello, F., Vassallo, G.: Ordering Web Pages through the Use of the Sammon Formula and the CGRD Algorithm. Proc. of AICA Congress, 27-30 Oct, 2000 Taormina (ME) -Italy - pp.495-503 449, 454

[4] Honkela, T., Leinonen, T., Lonka, K., Raike, A.: Self-Organizing Maps and Constructive Learning. Proc. of ICEUT'2000, Beijing, August 21-25, pp. 339-343 449

[5] Siolas, G., d'Alche-Buc, F.: Support Vector Machines based on a semantic kernel for text categorization. Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, IJCNN 2000, Volume: 5 , 2000 Page(s): 205 -209 vol.5 449

[6] Yang, H., Lee, C.: Automatic category generation for text documents by self-organizing maps. Proc. of IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000, Vol.3 , 2000 pp: 581 -586 449

[7] Siivola, V.: Language modeling based on neural clustering of words. IDIAP-Com 02, Martigny, Switzerland, 2000. 449

[8] Honkela, T., Pulkki, V., Kohonen, T.: Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map. Proceedings of International Conference on Artificial Neural Networks, ICANN-95, F. Fogelman - Soulie and P. Gallinari (eds.), EC2 et Cie, (Paris, 1995) 3-7. 449, 459

[9] Landauer, T. K., Foltz, P. W., Laham, D.: Introduction to Latent Semantic Analysis Discourse Processes, 25, pp: 259-284. 1998 450, 459

[10] Burgess, C., Lund, K.: The Dynamics of Meaning in Memory Cognitive dynamics: Conceptual and Representational Change in Humans and Machines. E. Dietrich and A. Markman (Eds.), Hillsdale, N. J.: Lawrence Erlbaum Associates, Inc., 2000 450, 459

[11] Sahlgren, M., Karlgren, J., Cöster, R., Järvinen, T.: SICS at CLEF 2002: Automatic Query Expansion Using Random Indexing The CLEF 2002 Workshop, September 19-20, 2002, Rome, Italy. 450, 459

[12] Steyvers, M., Shiffrin, R. M., Nelson, D. L.: Semantic spaces based on free association that predict memory performance http://www-psych.stanford.edu/ msteyver/papers/DIssertationNewA.pdf. 450, 459

[13] Levy, J. P., Bullinaria, J. A.: Learning Lexical Properties from Word Usage Patterns: Which Context Words Should be Used? , Connectionist Models of Learning, Development and Evolution: Proceedings of the Sixth Neural Computation and Psychology Workshop, R. F. French and J. P. Sougne (Eds), pp: 273-282. London: Springer, 2001 450, 459

[14] Widdows, D., Cederberg, S., Dorow, B.: Visualisation Techniques for Analysing Meaning Fifth International Conference on Text, Speech and Dialogue, Brno, Czech Republic, September 2002, pp 107-115. 450, 459

[15] Magnini, B., Strapparava, C.: Experiments in word domain disambiguation for parallel texts Proc. of SIGLEX Workshop on Word Senses and Multi-linguality, Hong-Kong, October 2000. held in conjunction with ACL2000. 450, 459

[16] Miller, G.A, Beckwidth, R., Fellbaum, C., Gross, D., Miller, K. J.: Introduction to WordNet: An On-line Lexical Database. International Journal of Lexicography, 1990, Vol. 3, No.4, 235-244. 450, 451, 455, 458

[17] Sammon, J. W. Jr.: A Nonlinear Mapping for Data Structure Analysis. IEEE Transactions on Computers, Vol. C-18, no. 5. (May 1969) 401-409 450, 452, 454, 455, 458

[18] Didion, J.: JWNL (Java WordNet Library). http://www.sourceforge.net 453
[19] Sloan, K. R. Jr., Tanimoto, S. L.: Progressive Refinement of Raster Images. IEEE Transactions on Computers, Volume 28, Number 11. (November 1979) 871-874 450, 454

# A Relation-Based Schema for Treebank Annotation

Cristina Bosco and Vincenzo Lombardo

Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
{bosco,vincenzo}@di.unito.it
http://www.di.unito.it/~tutreeb

**Abstract.** This paper presents a relation-based schema for treebank annotation, and its application in the development of a corpus of Italian sentences. The annotation schema keeps arguments and modifiers distinct and allows for an accurate representation of predicate-argument structure and subcategorization. The accuracy strongly depends on methods adopted for defining the relations which are tripartite feature structures that consist of a morpho-syntactic, a functional and a semantic component. We presents empirical evidence for these tripartite structures by illustrating phenomena faced in the development of an Italian treebank.

## 1 Introduction

Currently, most NLP systems that operate in realistic settings use statistical methods trained on *treebanks*, very large corpora of syntactically annotated sentences. The major existing treebanks are the Penn Treebank for English ([18, 19, 1]), the Prague Dependency Treebank for Czech ([12, 1]), and the NEGRA-TIGER project for German ([4, 1, 3, 10]. Moreover, there is a worldwide effort for the development of such a resource (see [1] for reports about Spanish, Italian, French, Chinese, Polish, Japanese and Turkish).

Treebanks can be viewed as text repositories where the implicit linguistic information is made explicit through the process of annotation. Linguistic information consists of Part Of Speech tagging, including syntactic categories and possibly other features (gender, number, tense, ...), bracketing structure, and *syntactic dependencies*. Syntactic dependencies are a relevant part of the annotation, since the level of representation that many applications actually need is a representation of predicate-argument structure which provides a useful interface to a semantic or conceptual representation [23].

The relevance of the predicate-argument structure has been clearly demonstrated by [14], referring to the Machine Translation task, where the contribution of the predicate-argument structure to the quality of the translation output exceeds the contribution of syntactic structure and vocabulary coverage (this has been further discussed in [15]). The syntax-semantics interface has been useful in Question Answering approaches that match the semantic roles in the question

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 462–473, 2003.
© Springer-Verlag Berlin Heidelberg 2003

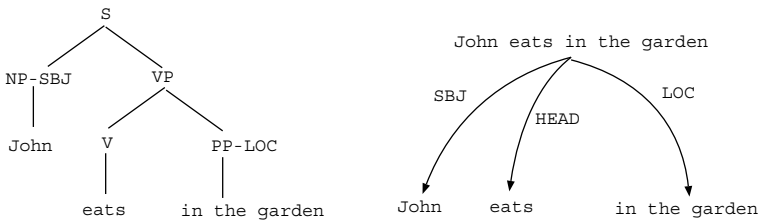



**Fig. 1.** An example in Penn-like format and a possible representation of the relational structure (SBJ=subject, LOC=location)

with the semantic roles of the (documents which can contain the) answer (see, e.g. the TREC-8 system described in [16]). The use of semantic roles too provides a great contribution to solving a number of phrase attachment ambiguities [6]. Considerable progress has been made in parsing by systems based on lexicalized probabilistic models that take into account syntactic dependencies (e.g. [9, 8]). Also, syntactic dependencies have been used successfully in NLP-based Information Extraction [27] and Information Retrieval [28].

All major approaches to treebank annotation include some forms of syntactic dependencies (see section 2), and the research for the enhancement of the annotation formats has led some projects to introduce further annotation levels, in propositional and pragmatic terms. However, there is not an agreement among the various proposals on the number and type of dependencies that are to be annotated in order to yield effective treebank-trained applications (see [13] for a comparison among several annotations of syntax-semantics information). Each approach comes with its own set of dependencies that is mostly guided by the actual data in the corpus at hand. Moreover, all the approaches miss to represent explicitly the interrelation among syntactic dependencies which is fundamental for the extraction of accurate linguistic knowledge and for the design of statistical models.

The goal of this paper is to systematize the representation of syntactic dependencies by providing a single-layered reference framework for treebank annotation. The reference framework relies on the notions of *grammatical relation* and *relational structure*, that have a long tradition in linguistics and are strongly related to the predicate-argument structure and the syntax-semantics interface (see figure 1 on the right).

These relational notions encode linguistic knowledge that is more proximate to semantics, and underlies syntax and morphology. Constraints on relations are expressed through subcategorization and valency, that inform about the lexical realization of a predicate, the number and type of arguments that a predicate requires, and the mapping from these syntactic arguments to a semantic representation [6]. The importance of the relational structure is testified by a number of formalisms: Relational Grammar [22] presents a relation-based sentence structure; Lexical Functional Grammar [5] introduces a relational struc-

ture, the f-structure, that is pivoted by the constituency structure; Categorial Grammar [17], forms a family of formalisms which represent the language in terms of functions and arguments encoding the syntax-semantics interface.

This paper proposes a single-layered relation-based schema for the representation of syntactic dependencies in treebank annotation. The focus of the paper is the representation of grammatical relations as tripartite feature structures that consist of a morpho-syntactic, a functional, and a semantic component. The schema is applied to the annotation of a corpus of Italian sentences, the Turin University Treebank (TUT), and we present evidence for the tripartite feature structure from the corpus sentences.

The paper is organized as follows: after a review of the representation of the syntactic dependencies in the three major approaches to treebank annotation, we motivate and introduce the tripartite structure of grammatical relations, the relational structure and the TUT annotation schema. Finally we provide empirical evidence for the tripartite feature structure from the TUT corpus.

## 2 Related Approaches

In this section we review the approaches to the annotation of syntactic dependencies in three known schemata, that have been successfully applied to the annotation of very large corpora.

The constituency-based Penn schema [18] is augmented with a limited number of "function" tags associated with constituents (see figure 1 on the left), and organized in three sets [19]: form/function tags, grammatical tags, and semantic role tags. Form/function tags assume a default morpho-syntactic function associated with each constituent type (form), and are used to mark those constituent labels for which the actual function is not the default function. For example, ADV (= adverbial) is a function tag that marks clauses and NPs that behave like adverbs, NOM (= nominal) marks non-NP that functions as NP. Grammatical tags refer to the relational structure in terms of syntactic functions: typical examples are SBJ (surface subject), LGS (logical subject in passives), DTV (dative object in unshifted dative constructions). Finally, semantic role tags are BNF (benefactive), DIR (direction), LOC (location), MNR (manner), ... . Each constituent can be tagged with multiple tags, but never with two function tags from the same set.

The goal of a more accurate annotation of the predicate-argument structure has triggered a novel project, known as PropBank [21, 15]. The idea is to develop a new treebank of propositional structures associated with sentences.

Negra combines the annotation of constituents and relations with a mixed representation of both the bracketing structure and the syntactic dependencies, possibly spanning long distances. The bracketing structure consists of words and phrases; in order to conveniently represent German free word order without introducing a large number of traces, it allows discontinuous constituents. Syntactic dependencies between the bracketing units are represented by special nodes: HD (head) is a specific label that marks the head word of a phrase, several

syntactic functions (DA = dative, JU = junctor, MNR = post-nominal modifier, MO = modifier, OC = clausal object, PD = predicative, SB = subject, SBP = subject in passives, ...).

A specific enhancement with respect to Penn annotation is some distinction between arguments and modifiers and the identification of the head of a phrase. However, Negra annotation schema lacks semantic roles, which are the object of the TIGER project [3, 26].

The annotation schema of Prague descends directly from the functional generative approach to dependency syntax [24]. The schema consists of three separate levels, morphological, analytical (i.e. functional-syntactic) and tectogrammatical (i.e. semantic/pragmatic). The morphological level consists in information concerning a single word. The analytical level connects the sentence words (phrases are banned in the dependency approach) with grammatical relations (called analytical functions - e.g., Pred (predicate), Sb (subject), Obj (Object), Adv (adverbial), Atv (complement depending on non-Verb), AtvV (Verbal complement), Atr (attribute), AuxC (conjunction), AuxV (auxiliary "to be"), AuxK (punctuation) Coord (coordination node), ...). The tectogrammatical level is structurally based on dependency syntax, and represents the syntax-semantics/pragmatics interface, in order to reveal the topic-focus articulation of the sentence. Here there are about 40 functors that represent semantic roles (actor/bearer, addressee, benefactive, origin, effect, cause, manner, locative, ...).

Abstracting from the three approaches reviewed above, we can individuate three major aspects in syntactic dependencies annotation: a morpho-syntactic aspect, a functional-syntactic aspect, and a semantic-syntactic aspect. These three aspects are conveyed by the notion of grammatical relation, a feature structure where the interrelations among the various dependencies can be conveniently represented.

## 3 A Relation-Based Schema for Treebank Annotation

In this section we introduce our relation-based approach to treebank annotation. Relations are represented as feature structures [25], partitioned in three components, morpho-syntactic, functional-syntactic, and semantic-syntactic. The set of all the relations in a sentence is the Augmented Relational Structure (ARS), that augments standard relational structure (in functional terms) with information about the morpho-syntactic realization of the relation and the mapping to a semantic role. The Augmented Relational Structure (ARS) is a directed acyclic graph (dag), where nodes are syntactic units of the sentence and edges are grammatical relations between the syntactic units. Syntactic units are strings of words of the sentence, possibly up to individual words. Now we introduce the three components, and then the ARS-based annotation schema.

### 3.1 The Morpho-Syntactic Component

The morpho-syntactic component is useful in making explicit morpho-syntactic variants in the realization of one predicate-argument structure, that is expres-

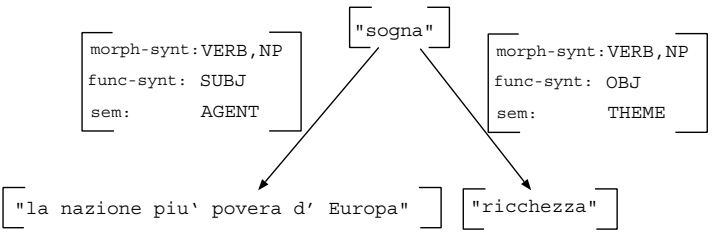



**Fig. 2.** A verbal realization of a predicate-argument structure: "La nazione più povera d'Europa sogna ricchezza" (The poorest nation of Europe dreams wealth)

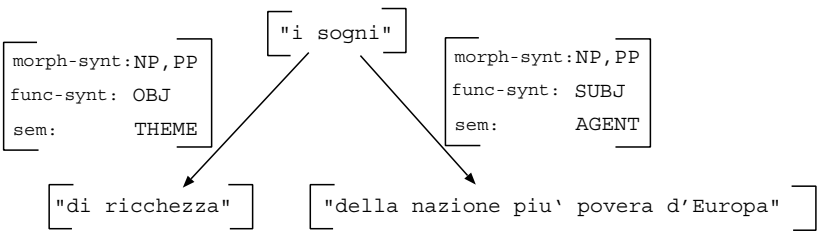



**Fig. 3.** A nominal realization of a predicate-argument structure: "I sogni di ricchezza della nazione più povera d'Europa" (The dreams of wealth of the poorest nation of Europe)

sions that differ for the morpho-syntactic features of at least one word (e.g., *correre velocemente* (*to run quickly*) is a morpho-syntactic variant of *correre in modo veloce* (*to run in a fast way*)). In marking the morpho-syntactic variants, we can account for the influences between the surface realizations of the several portions of a predicate-argument structure: this provides hints for the recognition of the same predicate-argument through several surface realizations [15]. In the figures 2 and 3 we find two morpho-syntactic variants of the predicate-argument structure "someone dreams of something". The fact that they only differ with respect to the morph-synt component immediately reveals they are two variants of the same predicative structure: when the head is verbal, the subject is a NP which agrees in number with the verb (**la nazione*[SING] *più povera d'Europa sognano*[PLUR]); when the head is nominal, the subject is realized by a PP introduced by "*di*" and agreement is not enforced (*i sogni*[PLUR] *di ricchezza della nazione*[SING] *più povera d'Europa*).

The morpho-syntactic component is also useful in cases when different morpho-syntactic categories fulfil the same syntactic function within a large syntactic unit, though not part of a predicate-argument structure. For instance, both Noun Phrases and Prepositional Phrase (beyond Adverbs) can function as typical Adverbial Modifiers within a sentence, like in the sentence "*Pat partì presto/quel giorno/in quel giorno*" (Pat departed early/that day/in that day). So, NP and PP are two morph-synt variants of Adverbials.

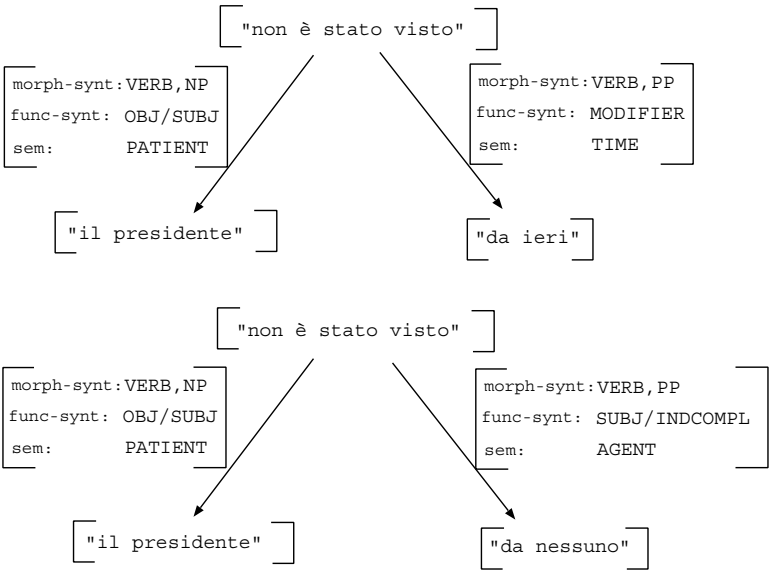



**Fig. 4.** The argument-modifier distinction in the functional-syntactic component

### 3.2 The Functional-Syntactic Component

The functional-syntactic component is the core of the grammatical relations, and refers to those relations (like Subject and Object) that have been considered as purely syntactic in the literature. For example, in Relational Grammar [22] and Lexical Functional Grammar [5] there are three arguments syntactically marked, i.e. Subject, Object, Indirect Object. The other arguments are often called 'indirect complements' and distinguished on a semantic basis.

While the morpho-syntactic component marks the differences among morpho-syntactic variants of one predicative structure, the functional-syntactic component is useful in marking the similarity among them. This component identifies the subcategorized elements, that is it keeps apart arguments and modifiers in the predicative structures. For instance, the structures in figure 4 are differentiated with respect to the func-synt component: in "*il presidente non è stato visto da ieri*" (the president has not been seen since yesterday), the preposition "*da*" introduces a modifier, whilst in "*il presidente non è stato visto da nessuno*" (the president has not been seen by nobody (i.e. nobody has seen the president)) it introduces an argument of the verb, the agent complement[1]).

[1] The X/Y notation refers to a relation X 'trasformed' into the relation Y. In the examples of fig. 4 the unit "il presidente" has been transformed from Object to Subject because of passivization. For details see linguitic notes at http://www.di.unito.it/~tutreeb/

The functions of the functional-syntactic component are organized in a hierarchy [2], as in various relational approaches to syntax (see, e.g., [11]). The hierarchical distribution of syntactic functions depends on criteria that define the degree of specification of a relation: the hierarchy proposed in this work distinguishes between Arguments and Modifiers. Arguments include Subject, Object, Indirect Complement, Predicative Complements (of the Subject and of the Object, respectively); modifiers are split into Restrictive Modifiers and Appositions. Syntactic functions also realize syntactically marked phenomena such as various forms of Coordination (correlative, comparative or adversative), Verbal Auxiliary functions (Tense, Progressive, Passive markers), Idiomatic utterance dependencies, Clausal Separators (through punctuation), Proper Name constructions. As it will be clear below, the hierarchy is useful when the annotator is not certain about some syntactic dependency and prefers to assign some less specific function.

### 3.3 The Semantic Component

The semantic component specifies the role of syntactic units in the syntax-semantics interface. It makes explicit the role that the participants play in the event described by the predicate (semantic or thematic roles such as Agent or Receiver), and supplementary information about the event (Location, Time, Manner, ...). Therefore it discriminates among different kinds of modifiers and arguments: for instance, in the following examples the functional-syntactic structure is the same, but the modifiers have different semantic roles (i.e. a measure of the amount of the wine or the material which the glass is made of, respectively).

*un bicchiere di vino* (*a glass of wine*)

*un bicchiere di cristallo* (*a glass of crystal, i.e. a crystal glass*)

The actual development of the semantic component in terms of the labels to be included in an annotation schema is a very hard task. We can identify at least three levels of generality: verb-specific roles (e.g. Runner, Killer, Bearer); thematic roles (e.g. Agent, Instrument, Experiencer, Theme, Patient); generalized roles (e.g. Actor and Undergoer). The use of specific roles can cause the loss of useful generalizations, whilst too generic roles do not describe with accuracy the data. The approach pursued in this work has been to annotate very specific semantic roles only when they are immediately and neatly distinguishable. Attributes like Age (e.g., "un uomo di 50 anni" - *a 50 year old man*) or semantic roles like TransportMean (e.g., "George venne in autobus" - *George came by bus*) are easily assigned; the semantic role of "senza metodi pesanti" (*without violent methods*) in the example of fig. 5 is not easy to assign and is left unspecified.

Now we move to the annotation schema.

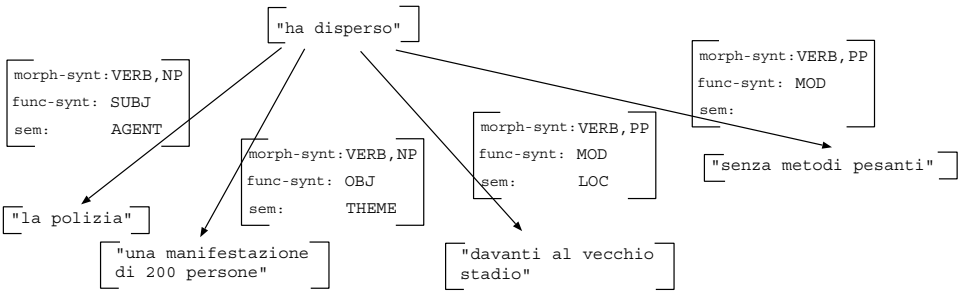



**Fig. 5.** A representation of the sentence "la polizia ha disperso una manifestazione di 200 persone davanti al vecchio stadio senza metodi violenti" (The police dispersed a demonstration of 200 people in front of the old stadium without violent methods)

### 3.4 The Augmented Relational Structure and the Annotation Schema

In this section we see how ARS has been used to develop a treebank annotation schema. First, we recall the major features of ARS and then we illustrate the implementation of the schema.

ARS is a single-layered structure that describes grammatical relations in terms of three separate components. This neat distinction of the components has not been an issue in the other treebank approaches described above, in the theoretical assumptions (Negra does not provide any partition of the functions), and in the annotation practice (Penn makes the theoretical distinction but annotation is practically restricted to a single function tag per dependency). In some approaches, the three components are viewed as distinct annotation layers (like in Prague). Even if the structure at the semantic level is not always isomorphic to the structure at the syntactic one (e.g., in quantifier scoping or coordination), in general, for phenomena that NLP applications take into account, we can represent semantic dependencies without going beyond the syntax-semantics interface. The reliability of such an approach is confirmed, for instance, by [23] which presents a syntax-semantic monostratal annotation, and by Prague too where both analytical and tectogrammatical layers share the same structure.

A consequence of a single-layered approach is that there are no problems of inter-layer alignment which must be solved in tasks involving more than one layer (e.g., PP-attachment in parsing), and which are usually hard to implement because of structural differences among independent levels. For instance, in Prague the inter-layers alignment is rather complex, because the number of nodes of the tectogrammatical (semantic) level is different from that at the analytical (syntactic) level. Finally, the tripartite structure of the relations in ARS guarantees that different components can be accessed and analyzed independently (like in [20]).

The ARS-based approach has been applied to the development of an Italian corpus, the Turin University Treebank (TUT, http://www.di.unito.it/~tutreeb/).

In the case of TUT, the implemented annotation schema is a dependency-based representation. In this schema the syntactic units are single words and the grammatical relations are dependency relations described as tripartite feature structures.

Given the ARS-based annotation schema, two are the issues at hand in the development of the treebank: what are the actual feature values in the three components that form the grammatical relations and how they scale when the corpus comes to a relevant size.

The choice of an adequate set of feature values for each component is a critical one. The use of a large variety of values provides a great accuracy and specialization in the description of relations, but contrasts with reasonable time in annotation, since annotators' task becomes very time-consuming for the assignment of the correct relation. This is due to two factors: first, the selection of the correct grammatical relation can be more difficult when the search space consists of a large number of competing labels; second, the specificity of relations can decrease the inter-annotator agreement. Finally, a great number of feature values requires for an application a very large set of training data, otherwise the problem of sparseness becomes relevant and the statistical models are not reliable. On the other hand, if we maintain the feature values at very general levels the annotation schema provides very few hints of the syntactic dependencies, and it is likely that statistical models trained on these data yield the same results in a large number of situations.

The solution in our approach is to adopt a variable degree of specificity, implemented in two ways: one is to drop some of the three components, the other is the hierarchical organization of the syntactic functions. Let us start from the first. All the relations annotated in TUT include the functional-syntactic component. This is the only component in the 23.1% of the relations, in case they represent purely syntactic dependencies like, e.g., idiomatic expressions or proper name compounds. The 35.2% of relations lacks a morpho-syntactic component. Only the 12.1% of relations include the semantic component. The motivations for such a low percentage of semantically annotated relations are: first, not all the relations carry semantic information (in fact, some relations are purely syntactic, see above, and typically the relations linking function words, like determiners or prepositions, do not carry relevant semantic contribution and in the Prague treebank are pruned at the semantic level); second, at this stage of the treebank development, the semantic component is systematically annotated only on modifiers and indirect complements (exclusively distinguished on a semantic basis), but not on terms (subject, object and indirect object)[2]. The reference to ontological and/or lexico-semantic resources will further increase the consistency and accuracy of the semantic annotation ([21] and [20] use a database of predicate-argument structures).

The second implementation of variable degree of specificity relies on the hierarchy of functions, in that the annotator provides the most specific relation

[2] A consistent semantic annotation of terms, e.g., as agent, patient, is a future task we will perform by using a database of subcategorization frames.

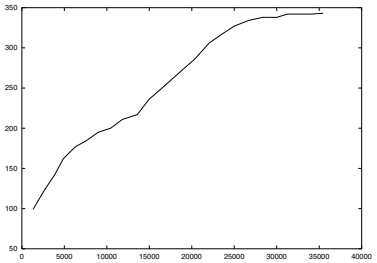



**Fig. 6.** Grow of grammatical relations for the 1,200 sentences

he/she feels confident about. If specific annotations result in a problem of data sparseness, the hierarchy can provide a controlled mechanism of backing-off.

The hierarchical organization also allows the treatment of vagueness, because the high levels of the hierarchy offer broad categories of relations (e.g., the distinction between arguments and modifiers, or between restrictive and non-restrictive modifiers). The hierarchy is again a way of dealing with ambiguity, when the annotator is not able to select the correct relation. For instance, in the theoretical example *"Mario non l'ha ancora visto Gianni"* (*Mario has not seen Gianni yet/ Gianni has not seen Mario yet*), both *Mario* and *Gianni* can be subject or object [7]. In this case, the annotators can use the hierarchy in the bottom-top direction, from specific to generic levels to find a common parent of the involved relations. Other treebank annotation schemata offer diferent solutions: Prague assigns a second label or adds a special marker that denotes the uncertainty of the assignment; Penn uses a default anonymous constituent label X (when the bracketing is certain, but the label is uncertain) and the so-called *pseudo-attachment* (for structures that are globally ambiguous as attachment).

In order to see how this annotation schema scales over corpora of large sizes, we provide some data on the annotation of the TUT. The TUT consists of newspaper articles (about 60%), civil code (legal language - about 30%), miscellaneous (about 10%). Currently, we have annotated 1,200 sentences (about 35,000 words) with the current numbers of feature values: the morpho-syntactic component includes 40 items, the functional-syntactic component includes 71 items, the semantic-syntactic component includes 102 items. These 213 items combined into 343 valid feature structures (grammatical relations). This means that only few combinations are meaningful. All the three components seem to be very stabilized, and although it is preliminary to assert anything about the behavior of the same components in a very large corpus, the grow of the feature-values set have been very limited in the sentences from 950 to 1,200 (see fig. 6), and this is very promising for the future.

## 4 Conclusion

The paper presented a schema for treebank annotation based on an Augmented Relational Structure. The ARS is a representation of syntactic dependencies (aka grammatical relations) as tripartite feature structures, namely complex objects which take into account various interrelated informational domains, called components (morpho-syntactic, functional-syntactic and semantic-syntactic).

The paper has illustrated the three components in detail, and has provided some quantitative data on the annotation of the Turin University Treebank.

## References

[1] Abeillé, A. (ed.): Building and using syntactically annotated corpora. Kluwer, Dordrecht (2003) 462
[2] Bosco, C.: Grammatical relation's system in treebank annotation. In: Miltsakaki, E., Monz, C., Ribeiro, A. (eds.): Proceedings of Student Research Workshop of ACL/EACL, Toulouse France (2001) 1–6 468
[3] Brants, S., Dipper, S., Hansen, S., Lezius, W., Smith, G.: The TIGER Treebank. Proceedings of the Workshop on Treebanks and Linguistic Theories, Sozopol Bulgaria (2002) 462, 465
[4] Brants, T., Skut, W., Uszkoreit, H.: Syntactic annotation of a German newspaper corpus. Proceedings of Treebanks workshop - Journées ATALA sur les corpus annotés pour la syntaxe, Paris (1999) 69–76 462
[5] Bresnan, J. (ed.): The mental representation of grammatical relations. MIT Press, Cambridge, Mass (1982) 463, 467
[6] Briscoe, T.: From dictionary to corpus to self-organizing dictionary: learning valency associations in the face of variation and change. Proceedings of Corpus Linguistics 2001, Lancaster UK (2001) 79–89 463
[7] Carrol, J., Briscoe, E., Sanfilippo, A.: Parser evaluation: a survey and a new proposal. Proceedings of LREC98, Granada, Spain (1998) 447–454 471
[8] Charniak, E.: A maximum entropy inspired parser. Proceedings of the First NAACL, Seattle WA (2000) 132–139 463
[9] Collins, M.: Head-driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania, http://www.ai.mit.edu/people/mcollins/ (1999) 463
[10] Dipper, S., Brants, T., Lezius, W., Plaehn, O., Smith, G.: The TIGER treebank. Proceedings of LINC'01, Leuven Belgium (2001) 462
[11] N. M. Fraser, R. A. Hudson: Inheritance in Word Grammar, Computational Linguistics, Vol. 18, Num. 2 (1992) 133–158 468
[12] Haijcovà, E.: Dependency-based underlying-structure tagging of a very large Czech corpus. In: Kahane, S. (ed.): Traitement automatique de langues. Vol. 41, Num. 1 (2000) 57–78 462
[13] Haijčová, E., Kučerová, I.: Argument/valency structure in Propbank, LCS database and Prague Dependency Treebank: a comparative study. Proceedings of LREC 2002, Las Palmas Spain (2002) 846–851 463
[14] Han, C., Lavoie, B., Palmer, M., Rambow, O., Kittredge, R., Korelsky, T., Kim, N., Kim, M.: Handling Structural Divergences and Recovering Dropped Arguments in a Korean/English Machine Translation System. Proceedings of the Association for Machine Translation in the Americas 2000, LNAI Springer Verlag (2000) 462

[15] Kinsbury, P., Palmer, M.: From TreeBank to PropBank. Proceedings of LREC 2002, Las Palmas Spain (2002) 462, 464, 466
[16] Litkowski, K. C.: Question-Answering Using Semantic Relation Triples. Proceedings of TREC-8. NIST, Gaithersburg MD (1999) 349–356 463
[17] MacGee Wood, M.: Categorial grammar. Routledge (1993) 464
[18] Marcus, M. P., Santorini, B., Marcinkiewicz, M. A.: Building a Large Annotated Corpus of English. The Penn Treebank. Computational Linguistics, Vol. 19 (1993) 313–330 462, 464
[19] Marcus, M. P., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., Schasberger, B.: The Penn Treebank: Annotating Predicate Argument Structure. Proceedings of The Human Language Technology Workshop, Morgan-Kaufmann, San Francisco (1994) 462, 464
[20] Montemagni, S., Barsotti, F., Battista, M., Calzolari, N., Corazzari, O., Lenci, A., Zampolli, A., Fanciulli, F., Massetani, M., Raffaelli, R., Basili, R., Pazienza, M. T., Saracino, D., Zanzotto, F., Mana, N., Pianesi, F., Delmonte, R.: Building the Italian Syntactic-Semantic Treebank. In: Abeill'e, A. (ed.): Building and using parsing corpora, Kluwer Dordrecht (2003) 189–210 469, 470
[21] Palmer, M., Dang, H. T., Rosenzweig, J.: Semantic tagging for the Penn Treebank. Proceedings of LREC 2000, Athens Greece (2000) 699–704 464, 470
[22] Perlmutter, D. M.: Studies in Relational Grammar 1. University of Chicago Press, Chicago (1983) 463, 467
[23] Rambow, O., Creswell, C., Szekely, R., Taber, H., Walker, M.: A dependency treebank for English. Proceedings of LREC 2000, Athens Greece (2000) 857–863 462, 469
[24] Sgall, P., Haijcova, E., Panevova, J.: The meaning of the sentence in its pragmatic aspects. Reidel Publishing Company (1986) 465
[25] Shieber, S. M.: An introduction to unification-based approaches to grammar. CSLI, Stanford (1986) 465
[26] Smith, G.: Encoding thematic roles via syntactic functions in a German treebank. Proceedings of workshop on syntactic annotation of electronic corpora. Tübingen, Germany (2002) 465
[27] Vilain, M.: Inferential Information Extraction. In: Pazienza, M. T. (ed.): Information Extraction. LNAI 1714, Springer (1999) 95–119. 463
[28] Vilares, J., Barcala, F. M., Alonso, M. A.: Using syntactic dependency-pairs to improve retrieval performance in Spanish. Proceedings of CICLing-2002 (2002) 381–390 463

# Personalized Recommendation of TV Programs*

L. Ardissono[1], C. Gena[1], P. Torasso[1], F. Bellifemine[2],
A. Chiarotto[2], A. Difino[2], and B. Negro[2]

[1] Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, Torino, Italy
{liliana,cgena,torasso}@di.unito.it
[2] Telecom Italia Lab, Multimedia Division
via Reiss Romoli 274, Torino, Italy
{bellifemine,chiarotto,difino,barbara.negro}@tilab.it

**Abstract.** This paper presents the recommendation techniques applied in Personal Program Guide (PPG), a system generating personalized Electronic Program Guides for digital TV. The PPG recommends TV programs by relying on the integration of heterogeneous user modeling techniques.

## 1 Introduction

The advent of Internet and Word Wide Web makes now available to the users a large amount of information, products and services. Recommendation techniques [13] based on the exploitation of AI techniques such as user modeling, content based and collaborative filtering are thus often presented as a solution to the information overload problem by helping the users to filter relevant items on the basis of their needs and preferences. With the recent expansion of TV content, digital TV networks and broadband, smarter TV entertainment is needed as well. As there are several hundreds of available programs every day, users need to easily find the interesting ones and watch such programs at the preferred time of day. Electronic Program Guides (EPGs) should recommend personalized listings, but they should also be deeply integrated in the TV appliance, in order to facilitate the access to the user's digital archive. For details, see [2].

This paper presents Personal Program Guide, a user-adaptive EPG that tailors the recommendation of TV programs to the viewer's interests, taking several factors into account. The PPG captures an individual model for each registered user and employs it to generate an EPG whose content and layout are tailored to the user watching TV.[1] The personalized recommendation of programs is

* This work was partially supported by the Italian M.I.U.R. (Ministero dell'Istruzione dell'Università e della Ricerca) through the Te.S.C.He.T. Project (Technology System for Cultural Heritage in Tourism). We are grateful to Flavio Portis, who helped us in the development of the Stereotypical UM Expert of the PPG.

[1] At the current stage, we have focused on the personalization of the EPG to individual TV viewers. The management of household viewing preferences is part of our future work.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 474–486, 2003.
© Springer-Verlag Berlin Heidelberg 2003

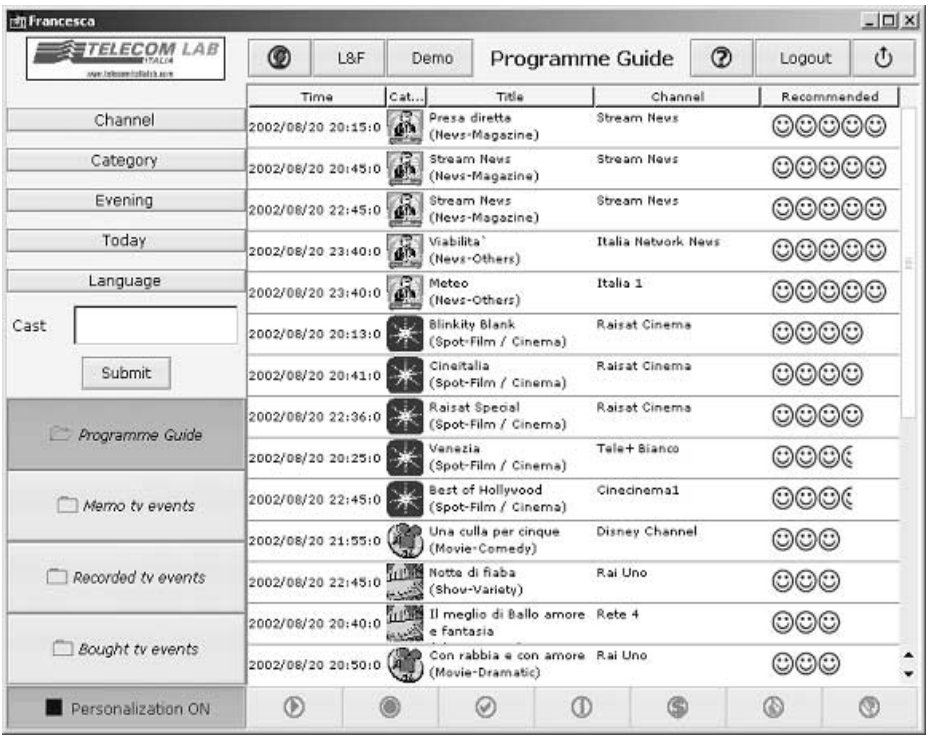



**Fig. 1.** Personal Program Guide: PC simulator main window

based on the integration of user modeling techniques relying on explicit user preferences, stereotypical information about TV viewer preferences, and the unobtrusive observation of the user's viewing habits. As the PPG has been designed to run within a Set-Top box, the user's behavior can be continuously monitored, in contrast to the Web-based EPGs, which can only track the interaction while the user browses them. In particular, we preferred to focus on the observation of real user's behavior and to implement the PPG on the client- side. Basing the system on a server-side architecture would support the application of social recommendation techniques, such as collaborative filtering, but would waste the rich information coming from the direct observation of the user's behavior.

In the following, Section 2 gives an overview of the facilities offered by the system, Section 3 sketches the system architecture and then faces with the management of the sources of information about the users, Section 4 discusses the recommendations of TV programs, Section 5 describes the results of an evaluation, Section 6 presents related works and Section 7 concludes the paper.

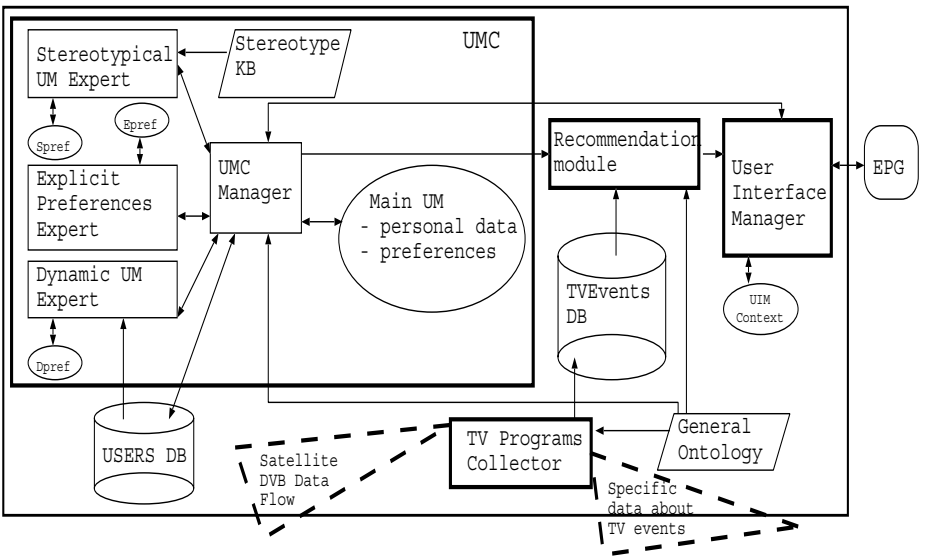



**Fig. 2.** Architecture of the Personal Program Guide

## 2 The Facilities Offered by the PPG System

The PPG, which acts as a personal assistant offering advanced TV services, is designed for a set-top box, but it is currently implemented in a simulator running on desktop environments for demonstration purposes. In order to make our description more concrete, we will use an example of the GUI of this prototype: see Figure 1. The system offers advanced facilities for browsing the TV events: programs can be searched by channel, category, viewing time, etc.. Moreover, the user may ask for details about a program (e.g., cast, content description), record it, ask to be advised when the transmission of the program starts (Memo TV events button), and so forth. The archived programs are retrieved by means of functions that enable the user to get the list of programs she wishes to be alerted about, she has recorded (Recorded TV Events), or she has bought (Bought TV Events). By default, the PPG works in personalized mode (Personalization ON): the less suitable programs are filtered out and the most promising ones are shown at the top of the recommendation list. The recommendation degree of a program is represented by a list of smiling faces close to the program description, in order to make the ranking information independent of the visualization criterion (time, channel, etc.). The personalization facility can be switched off by the user.

## 3 Overview of the System Architecture

The architecture of the PPG is sketched in Fig. 2. In particular, the Recommendation Module makes use of the information about TV programs and preferences

of the users (managed by the UMC Manager) to generate the recommendation lists for the user.

The representation of TV programs is an extension of the Digital Video Broadcasting standard [6].[2] Each program is described by a set of fields specifying data such as the starting time of the program, the transmission channel and the stream content (video, audio or data). The descriptor also includes one or more program categories representing the program content and format: e.g., Movie, Serial, News; see [4]. The program categories are organized in a taxonomy, the General Ontology, which includes several broad categories, such as Serial, and specializes such categories into more specific ones, such as Soap Opera, and Science Fiction Serial. The management of the user model is aimed at achieving a precise description of her interests and viewing preferences, during different times of day (and weekdays). In the design of the user model (UM), we considered the following information:

- *Explicit preferences* for categories of TV programs (e.g., movies, documentaries, etc.) that the user may want to notify the system about.
- The *estimates on the viewing preferences* for the program categories (related to the number of programs the user watches, for each category).
- *Socio-demographic information,* such as her age, occupation, and so forth.
- Information about the user's general *interests and hobbies.*
- *Prior information about the preferences of stereotypical TV viewers.*

Such different types of information provide multiple points of view on the user, useful for personalization, but require a separate information management. To this purpose, we have designed the User Modeling Component of the PPG as an agent that exploits three specialized user modeling modules (Explicit Preferences Expert, Stereotypical UM Expert, Dynamic UM Expert), each one managing a separate user model that reflects the viewpoint of the module (see Figure 2):

- The *Explicit User Model* stores the information elicited from the user in an explicit way: personal data, interests, and preferences for TV program categories.
- The *Stereotypical User Model* stores the prediction on the user's preferences inferred by exploiting general information about TV viewer categories.
- The *Dynamic User Model* stores the system's estimates on the user's preferences, as observed by analyzing the individual user's viewing behavior .

Each expert manages a different TV program ontology depending on the information about user preferences available to the expert. Then, mapping rules are applied to relate the different TV program characterizations to the categories of the General Ontology.

The User Modeling Component (UMC) maintains a Main User Model as a synthesis of such views, used by the system to personalize the interaction with the TV viewer. The UMC integrates the predictions provided by the experts

[2] The DVB has been defined at the international level to specify standards for the global delivery of digital television and data services.

Classification data
Age *[personal data]* *Importance*: 1, *Values*: (< 15, 0) (15/24, 0) (25/34, 0) (35/44, 0.5) (45/54, 0.5) (55/64, 0) (> 64, 0)
Gender *[personal data]* *Importance*: 1, *Values*: (male, 0) (female, 1.0)
Books *[interest]* *Importance*: 0.6, *Values*: (low, 0.8) (medium, 0.2) (high, 0)
Prediction part
*movies-sentimental*, Interest degree: 1; *serial-soap*, Interest d.: 1; *TV news* Interest d.: 0,2; *fashion programs*, Interest d.: 0,5; *cooking program*, Interest d.: 1, etc;

**Fig. 3.** The "Housewife" stereotype

into the Main User Model by taking the experts' confidence in the predictions into account (the confidence is based on the estimation of the quality of the data used to generate the prediction). For space reasons, we skip the description of the Explicit User Model, which manages in a direct way the explicit user's preferences and interests, and we focus on the other two user models, where inference mechanisms play a major role.

### 3.1 The Stereotypical User Model

We used the Sinottica lifestyle study conducted by Eurisko data analyzers [7] as a basis for specifying the characteristics and preferences of stereotypical TV viewer classes. Since the Eurisko survey relates homogeneous group of users and their corresponding interests and preferences, we structured the stereotypes in two parts: *i)* classification data characterizing the individuals of the represented stereotype, and *ii)* prediction part, containing the typical preferences of such individuals. Regarding the prediction part of stereotypes, we further analyzed a survey on the exposure to the TV, made by Eurisko and Auditel [1], which measures the audience of each lifestyles class. For more details about the knowledge engineering approach applied to collect and to process all the gathered data, see [8]. We defined a *Stereotype Ontology* defining the TV program categories to be considered as far as the stereotypical preferences are concerned. Mapping rules relate the preferences of the Stereotype Ontology to those ones of the General Ontology. The representation of the stereotypes is the one adopted in the SeTA system [3]; see the Housewife lifestyle in Fig. 3.

Each classification datum (socio-demographic feature, user interest) is represented as a slot with three facets: the *Feature Name*, the *Importance* (relevance of the feature to the description of the stereotype) and the *Values* (a frequency distribution on the values of the feature). For instance, the interest for *Books* has medium importance to the characterization of the users of the "Housewife" class (Importance is 0.6). Moreover, 80% of the "housewives" have low interest in reading books (frequency is 0.8); some have medium interest (0.2), but no one is highly interested in this activity.

The slots in the *prediction part* describe the preferences (for categories of the Stereotype Ontology) of the typical user belonging to the represented stereotype.

| Preferences for TV program categories | | Degrees of matching with stereotypes | |
|---|---|---|---|
| Movie-All 0.65 | conf: 0.43[4] | Colleagues | 0.34 |
| Movie-Sentimental 0.65 | conf: 0.43 | Engaged women | 0.24 |
| Movie-Comedy 0.65 | conf: 0.43 | Refined women | 0.26 |
| Movie-Detective 0.65 | conf: 0.43 | Dolphins | 0.16 |
| News All 0.65 | conf: 0.43 | | |
| Serial Fiction 0.65 | conf: 0.43 | | |

**Fig. 4.** Portion of Francesca's Stereotypical User Model

A prediction slot is represented as follows: the *Program category* specifies the TV program category. The *Interest degree* represents the user's interest in the category and takes values in [0,1], where 0 denotes lack of interest and 1 is the maximum interest. E.g., the *Housewife* really likes sentimental movies, soap opera and cooking programs; she moderately likes fashion programs and she does not like TV news.

To estimate the user preferences, the Stereotypical UM Expert first classifies the user with respect to the stereotypical TV viewer classes. This is aimed at estimating which lifestyle descriptions are best suited to predict her preferences. The classification is performed by matching the user's classification data with the stereotypical descriptions, according to the approach described in [3]. The result of the classification is a *degree of matching* with respect to each stereotype: this is a number in [0, 1], where 1 denotes perfect matching (the user data perfectly match the classification of a stereotype), while 0 denotes complete mismatch.[3]

Given the user's stereotypical classification, the predictions on the user's preferences (Spref in Fig. 2) are estimated by taking the contribution of each stereotype into account, proportionally to the degree of matching associated to the stereotype. Let's consider a program category C and the stereotypes $\{S_1, \ldots, S_n\}$. The user's degree of interest in C (Interest_C) is evaluated by means of the following weighted sum:

$Interest_C = \sum_{i=1}^{n} DM_{Si} * Interest_C_{Si}$ $Interest_C_{Si}$ is the degree of interest in C predicted by a stereotype Si and $DM_{Si}$ is the degree of matching between the classification data of the user and Si. Fig. 3 shows the classification of a user Francesca and the stereotypical predictions on her preferences.

**Confidence in the Stereotypical Predictions.** The confidence in the stereotypical predictions depends on the confidence in the user classification that, in

[3] For each datum $F$ (e.g., "Age"), the suitability of the stereotype $S$ is captured by a compatibility value, evaluated by matching the value $v$ of $S$ specified by the user (e.g., "35/44") with the corresponding datum in $S$. This match depends on the frequency of users belonging to $S$ fitting the $v$ value (e.g., 50% "housewives" are between 35 and 44) and on the importance of $F$ in $S$. The degree of matching of the user with respect to the stereotype is then evaluated by combining the compatibility values of her classification data by means of a fuzzy AND operation; see [3] for details.

turn, depends on the amount of information about the user available at classification time and on "how stereotypical" is the user.

*Confidence in the user classification with respect to a stereotype.* The user's degree of matching with respect to a stereotype S is considered reliable if it is based on complete information about her classification data. The confidence in the classification is thus evaluated by considering the minimum and maximum degrees of matching the user might receive, if complete information about her were available:

- The lower bound of the degree of matching ($DM_{min}$) is evaluated by pessimistically assuming that, for each classification datum she has not specified, the user matches the value(s) less compatible with the stereotype. For instance, several values of Age in Housewife have a compatibility equal to 0 (see Fig. 3). Thus, the lower bound of the compatibility of Age is 0.
- The upper bound ($DM_{max}$) is evaluated by optimistically assuming that, for each missing classification datum, the user matches the most compatible value (0.5 for Age in Housewife).

The lower and upper bounds define the interval of admissible values for the matching degree ($DM$), given the user data: $DM_{min} \leq DM \leq DM_{max}$. The larger is the interval the lower is the confidence in the classification. Thus, the confidence can be evaluated as: $confS = 1 - (DM_{max} - DM_{min}/\delta_{max})$ where $\delta_{max}$ is the maximum distance between $DM_{max}$ and $DM_{min}$ and corresponds to the case where no classification datum is set.

*Confidence in the predictions on the user's preferences.* To evaluate the confidence in the predictions on the user's preferences, an overall assessment of the quality of the user classification is needed, which takes all the stereotypes into account. We noticed that the estimates on the user preferences are accurate if she matches few stereotypes, while the predictions downgrade if she losely matches many stereotypes. Thus, we evaluate such confidence by combining the average confidence in the stereotypical classification (Confstereotypes) with an evaluation of its focalization degree (Focus).

StereotypicaExpertConfidence = Confstereotypes * Focus

The focalization degree is derived from the evaluation of Shannon's entropy on the degree of matching of the stereotypes. Suppose that the $\{S_1, \ldots, S_n\}$ stereotypes receive the following matching degrees $\{DM_1, \ldots, DM_n\}$. Then, the entropy is:

$$Entropy = \sum_{i=1}^{n} -DM_i * log_2 DM_i$$

As the number of stereotypes is fixed, the entropy may be normalized in [0,1], thus obtaining a normalized entropy normEntropy. The focalization degree is:

$$Focus = 1 - normEntropy$$

The focalization degree is 0 when the entropy is maximum, i.e., the classification is extremely uncertain. In contrast, when a single stereotype matches the user, the focalization degree is 1. In turn, the confidence in the prediction is high when the classification relies on complete information about the user and is very focused.

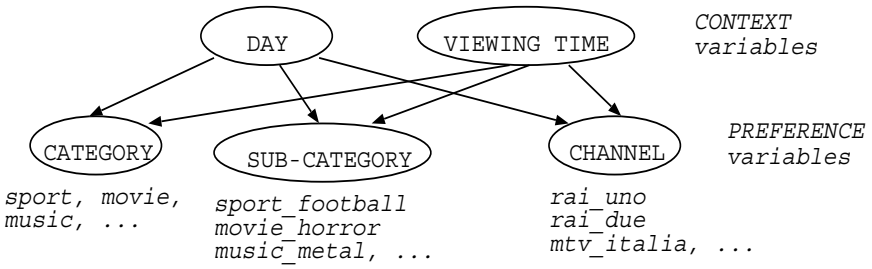



**Fig. 5.** Portion of the BBN that represents the Dynamic User Model

### 3.2 The Dynamic User Model

The Dynamic User Model specifies the user preferences for the program categories and subcategories of the General Ontology and for the TV channels available. Different from the other UM experts, the preferences can be related to various contexts because the expert has direct access to the user's behavior . In particular, this expert monitors user actions such as playing a TV program, recording a program and the like: see Fig. 1. In order to face the uncertainty in the interpretation of the user's viewing behavior, a probabilistic approach is adopted, where discrete random variables encode two types of information: preferences and contexts (viewing times). The sample space of the preference variables corresponds to the domain of objects on which the user holds preferences; the corresponding probability distributions represent a measure of such preferences (degrees of interest). The sample space of every context variable is the set of all possible contexts. We encoded this type of information by exploiting Bayesian Belief Networks (BBNs). Fig. 5 shows the structure of the BBN representing the user preferences. The network models the contextual information by means of context variables representing the conditions in which the user preferences for the TV programs may occur, the root nodes. We describe a context with temporal conditions, represented by the two variables "DAY" and "VIEWINGTIME" encoding, respectively, the 7 days of the week and the 5 intervals of time in which the day can be subdivided (morning, noon, ..., night). The leave nodes of the BBN represent the user's contextual preferences, providing the probabilities for every program category, subcategory and channel.

For each individual user, the BBN is initialized with a uniform distribution of probabilities on its nodes where all values assumed by the preference variables have equal probability. The BBN is updated by feeding it with evidence about the user's actions, starting from the first time she watches the TV. Each time the user interacts with the program guide (to record a TV program, play it, etc.), the category and the subcategory of the event and its transmission channel are retrieved. Then, the BBN is fed with evidence that a new observation for that category is available. Not all the user actions have the same impact on the learning phase: e.g., playing a TV program provides more important evidence about the user's preferences than asking for more information about the same

TV program. This fact is reflected in the definition of different learning rates for the possible user actions.[4]

**Confidence in the Predictions of the Dynamic UM Expert.** The confidence of the Dynamic UM Expert in the predictions on the user's preferences is based on the amount of evidence about the user's viewing behaviour provided to the BBN since the first user interaction. At the beginning, the expert has low confidence in its predictions. As the number of observations increases, the Dynamic UM Expert becomes more confident. In fact, although noise can be present in the user's behaviour, the BBN tolerates such noise much more in the presence of a large corpus of data. A sigmoid function is used to define the confidence, given the number of events observed within the context. This function is normalized in the interval [0,1] and is defined as follows:

$$Conf(x) = 1/(1+e)^{(k-x)*s}$$

The function returns a confidence close to 0 if no user-events are observed in a specific context. The function returns a confidence of 0.5 after k events are observed and the confidence gets close to 1 after the observation of 2*k user-events. The s coefficient (in [0, 1]) defines how steep has to be the function (s has been set to 0.1).

### 3.3 Integration of the Predictions Provided by the User Modeling Experts

The Main User Model is instantiated by merging the predictions on the user's preferences provided by the Explicit, Stereotypical and Dynamic UM Experts. For each preference P, the predictions on P ($Interest_1, \ldots, Interest_3$) are combined into an overall Interest as follows:[5]

$$Interest = [\sum_{e=1}^{n} Conf_e * Interest_e]/[\sum_{e=1}^{n} Interest_e]$$

The formula merges the predictions in a weighted way, on the basis of the experts' confidence, in order to privilege estimates based on higher quality information about the user. The confidence may change along time and eventually, the Dynamic UM Expert influences the predictions in the strongest way, providing an estimation of the user's long-term viewing preferences.

[4] The BBN exploited has been implemented using the Norsis' toolkit [12], which provides algorithms for general probabilistic inference and parametric learning.

[5] The Explicit and Stereotypical UM experts do not predict the preferred channels; thus, the confidence in such predictions is 0. Moreover, we assume that their (a-contextual) predictions are the same in all viewing contexts.

## 4 Personalized Recommendation of TV Programs

The Recommendation Module (Fig. 2) suggests TV programs as follows:

- the TV programs satisfying the user's search query are retrieved from a local database storing the information about available programs;[6]
- the programs are ranked on the basis of the user's preferences (Main User Model). The score associated to each the program is used to sort the recommendation list and to enrich the presentation with smiling faces representing the expected degree of appreciation;
- if several programs are retrieved from the user query, the rank associated to the items is exploited to filter out the worst programs, therefore reducing the length of the recommendation list.

Indeed, the generation of the scores for the TV programs is performed by taking into account both the user's preferences for the program category of the event and her preference for the transmission channel. The former type of information is the basis for the recommendations, but we use the latter to refine the score associated to the items, and thus the system's suggestions, with evidence about the user's viewing habits. The integration of these information sources is useful because the preferences for program categories do not support the comparison between individual programs belonging to the same category. In contrast, the preference for the channel (per viewing time) enables the system to take the user's preferences for individual programs into account, without explicitly modeling the characteristics of such programs. In fact, the system relies on the criteria applied by the provider in the selection of the programs to be shown: the scheduling of palimpsest (and of advertisements) is based on the supposed TV audience in a given time slot that influences the quality and the characteristics of the programs.

## 5 The Evaluation of the System's Recommendation Capability

In this initial phase of the project we carried out a formative evaluation of the system's recommendation capability. As the Dynamic UM Expert's predictions are not available at moment we focused on the other experts. 62 subjects, 22-62 aged and representing all the possible types of PPG users, have been interviewed by means of a questionnaire. To obtain the desired information, we collected the following data: general questions, information about general user's interests (books, music, sport, etc.) and preferences for categories (Movies, News, etc.) and subcategories (Action Movies, Cooking Programs, etc.) of TV programs.

[6] The local database is populated by the TV Program Collector that downloads from the MPEG-2 satellite stream information about the TV programs available in a restricted time interval and integrates such information with data retrieved from the providers' web sites.

The final questionnaire was made up of 35 questions where both the questions and the answers were fixed. The questionnaires were auto-filled by the users to avoid any possible interviewer's interferences and gained some days after the distribution. The questionnaire was anonymous and introduced by a written presentation explaining the general research aims. For the items concerning general data, participants were required to check the appropriate answer from a set of given answers. In the other questions, users had to express their level of agreement with the options concerning the given questions by choosing an item of a 3-point Likert scale. Then, the collected information has been entered into the system to evaluate the validity of the user classification and the accuracy of the recommendations. Concerning the first point, we have compared the system classification with the classification of two domain (human) experts. The comparison showed that 70% of the users have been correctly classified by the system, while the remaining 30% have been incorrectly classified for two main reasons:

- the system classification fails when the user's interests are different from those evaluated according her socio-demographic data;
- the data provided by Eurisko does not cover the whole Italian population.

The TV program predictions generated by the system have been then compared to the explicit preferences expressed by the users. As outlined before, this testing has mainly evaluated the recommendations provided by the Stereotypical UM expert in conjunction with the Explicit UM expert. In this case, when the user's explicit preferences are not available, the system takes in consideration just the Stereotypical expert. This situation is not unusual, since it is well known [5] that users do not like spend time filling questionnaires and evaluating items because they would get their tasks done immediately. Moreover, users are usually uncomfortable in answering personal questions.

To test the performance of the system, we evaluated the distance between the system predictions and the users' preferences by means of mean absolute error (MAE,[7] for details see [9]). In addition, to test the accuracy of the selection process we measured the *precision* of the collected data.[8]

We obtained a mean absolute error value of 0,10 (the values are expressed on a scale ranging from 0 to 1) with a precision of 0.50. These values have confirmed our hypothesis about the validity of an integration of different sources of information. We believe that the contribution of the Dynamic UM Expert and a broader coverage of the stereotypical KB can still improve these measures.

[7] The local database is populated by the TV Program Collector that downloads from the MPEG-2 satellite stream information about the TV programs available in a restricted time interval and integrates such information with data retrieved from the providers' web sites.

[8] The precision is defined as ratio between the user-relevant contents and the contents presented to the user.

## 6 Related Work

Several recommender systems are exploited in Web stores, electronic libraries and TV listing services; e.g., see [13], [10] and [11]. The PPG differs from such systems in two main aspects: our system integrates multiple acquisition techniques for the identification of the user preferences and the consequent recommendation of items. Moreover, the system privileges the local execution of tasks with respect to a centralized management of the EPG. In particular, the decentralization of the system execution supports the generation of precise user models (the TV viewers are frequent users of the TV) and limits the amount of explicit feedback required from the user, because her behavior can be analyzed at any time she watches the TV. In contrast, if a central server manages the EPG, the user's interaction with the TV is carried out in a distinct thread and can only be monitored while she browsers the program guide, unless special hardware is used to connect the TV to the internet in a continuous way. See, for instance, [14].

Some recommender systems integrate multiple prediction methods by evaluating their precision, given the user's reactions to the system's recommendations (programs she watches, etc.). For instance, Buczak et al. (see [2]) fuse three recommenders by means of a neural network. In contrast, the PPG currently merges the predictions provided by different UM Experts on the basis of their confidence in the predictions. Indeed, we want to exploit relevance feedback to fuse our UM Experts, as well. However, we will combine such feedback with the experts' confidence, because in this way the system can benefit from an informed tuning parameter during its whole lifecycle. In fact, as the confidence depends on the amount of information about the user available to the system, it can be employed since the first interaction, while relevance feedback takes a significant amount of time before being effective.

## 7 Conclusions

This paper has presented the Personal Program Guide (PPG), a prototype system generating personalized EPGs, which we are developing in a joint project between Telecom Italia Lab and the University of Torino. A demonstrator of the PPG running on a PC simulator of the Set-Top Box environment is available.

The PPG integrates different user modeling techniques for the recognition of the TV viewer's preferences and the consequent generation of personalized recommendation listings. The management of multiple perspectives on the recognition of the user's preferences, and the cooperation/competition between the different user modeling methods has revealed to be fruitful to enhance the system's recommendation capabilities. As expected, the personalization based only on stereotypical suggestions is problematic, because not always people match stereotypes in a precise way. At the same time, the recommendations based on explicit user information are subject to failures: users often refuse to declare their real preferences or they provide the system with weak information

about themselves. Finally, the recommendations based on the observation of the user behavior suffer from the *cold start* problem and, mirroring the user's usual selections, do not support the variety in the system's recommendations. The integration of three (or more) user modeling techniques enhances the reliability and richness of the system's predictions.

## References


[1] Auditel. http://auditel.it, 2000. 478

[2] L. Ardissono and A. Buczak. http://www.di.unito.it/ ˜ liliana/tv02/. In *Proc. of TV'02: the 2nd workshop on personalization in future TV*, Malaga, Spain, 2002. 474, 485

[3] L. Ardissono and A. Goy. Tailoring the interaction with users in Web stores. *User Modeling and User-Adapted Interaction*, 10(4):251–303, 2000. 478, 479

[4] L. Ardissono, F. Portis, P. Torasso, F. Bellifemine, A. Chiarotto, and A. Difino. Architecture of a system for the generation of personalized Electronic Program Guides. In *Proc. of the UM2001 Workshop on Personalization in Future TV*, Sonthofen, Germany, 2001. 477

[5] J. M. Carroll and M. B. Rosson. The paradox of the active user. In J. M. Carroll, editor, *Interfacing thought: cognitive aspects of Human-Computer interaction*, pages 80–111. MIT Press, Cambridge, MA, 1987. 484

[6] DVB. Digital video broadcasting. http://www.dvb.org, 2000. 477

[7] Eurisko. Sinottica. http://www.eurisko.it, 2000. 478

[8] C. Gena. Designing TV viewer stereotypes for an electronic program guide. In *Proc. 8th Int. Conf. on User Modeling*, pages 274–276, Sonthofen, Germany, 2001. 478

[9] N. Good, J. B. Shafer, J. A. Konstan, A. Botchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl. Combining collaborative filtering with with personal agents for better recommendations. In *Proc. 16th Conf. AAAI*, pages 439–446, 1999. 484

[10] GroupLens. Grouplens research. http://www.ncs.umn.edu/Research/GroupLens, 2002. 485

[11] A. Kobsa, J. Koenemann, and W. Pohl. Personalized hypermedia presentation techniques for improving online customer relationships. *The Knowledge Engineering Review*, 16(2):111–155, 2001. 485

[12] Norsys. Application for belief networks and influence diagrams, user's guide. http://www.norsys.com, 2001. 482

[13] P. Resnick and H. R. Varian, editors. *Special Issue on Recommender Systems*, volume 40. Communications of the ACM, 1997. 474, 485

[14] B. Smyth, P. Cotter, and J. Ryan. Evolving the personalized EPG - an alternative architecture for the delivery of DTV services. In *Proc. of the AH'02 Workshop on Personalization in Future TV*, pages 161–164, Malaga, 2002. 485

# A Simulation-Based Decision Support System for Forest Fire Fighting

Sung-Do Chi[1], Ye-Hwan Lim[1], Jong-Keun Lee[1], Jang-Se Lee[1], Soo-Chan Hwang[1], and Byung-Heum Song[2]

[1] Department of Computer Engineering, Hangkong University
200-1, Hwajon-dong, Deokyang-gu, Koyang-city
Kyonggi-do, 412-791, Korea
{sdchi,yahn,leejk,jslee2,schwang}@mail.hangkong.ac.kr
[2] Department of Flight Operation, Hangkong University
200-1, Hwajon-dong, Deokyang-gu, Koyang-city
Kyonggi-do, 412-791, Korea
{bhsong}@mail.hangkong.ac.kr

**Abstract.** The objective of this paper is to design and develop the efficient decision support system for forest managers that would help them in their decision-making tasks during the forest fire. To do this, we have adopted the advanced simulation techniques as well as the genetic algorithm for generating the forest fire fighting strategy. The GIS database with 3-D graphics has been also employed for supporting the decision-making. In order to coherently represent the geographical, meteorological, and forest information as well as to generate the fire model and simulation trajectory of the fire spread, the cellular modeling approach has been proposed. Various resources and their organizations model for the fire fighting can be efficiently represented by using the rule-based system entity structure and each resource organizations can be directly evaluated by employing the simulation-based genetic algorithm. Several simulation tests performed on a sample forest area will demonstrate our techniques.

## 1 Introduction

Wildfires burning in the forest are a serious issue for land managers and those living in or near rural areas. The earliest empirical studies identified the fuel type, fuel moisture, wind velocity, and slope as the predominant factors affecting the fire behavior [1]. Subsequent research has sought to quantify the relative importance of these factors and to produce tools for the prediction of wildfire behavior.

Models for the prediction of forest fire spread and fire behavior across landscapes have been available for many years [1]. The BEHAVE [2] of fire behavior prediction and fuel modeling was the first complete system that included computer programs. IGNITE [3] is a PC-based program that graphically presents the spread of fire across rasterised fuel cells as epidemic cell-to-cell propagation. This

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 487–499, 2003.
© Springer-Verlag Berlin Heidelberg 2003

technique easily caters for rapid spatial changes in fuel and topographic conditions; however, fire control authorities for operational use have not adopted it. Recently, the fire spread prediction systems such as CHARADE [4], SiroFire [2,4], have been implemented for supporting the strategic planning and resource allocation [5,6]. However, most systems employ their planning method based on the analytic technique rather than the simulation technique. That is, the regional fire models for the fire spread prediction should be coherently integrated with other related models such as firefighter, water pump, helicopter, etc. for representing and evaluating the planning strategies and resource allocation by simulation [7]. To allow this, we have proposed the cellular modeling approach in which the geographical, meteorological, and forest information as well as the fire spread model can be coherently represented and integrated. All possible resources can be easily represented by the rule-based system entity structure and evaluated by the simulation-based genetic algorithm. And the FOFIS has been developed by using the proposed methodology for managing forest fire.

This paper is organized as follows: First, it introduces the overall concept of FOFIS and then describes the cellular modeling and simulation approach for the forest fire spreading. Then, it proposes the automated methodology for generating forest fire fighting strategy. It is followed by the description of a case study.

## 2 System Overview

In this section we shall address the operational context in which the system will be deployed. The use of the system may be the controller based in a provincial fire station as shown in Fig. 1. There are the input data such as the first firing spot and the first sphere of fire influence, etc., detected by various kind of sensors such as forestry officer, fire management helicopter and so on. Also, it requires the additional input information such as the forest, meteorological and geographical information as well as the resource information; those maybe obtained from the forestry offices, meteorological departments, geographic centers, and/or fire departments, military offices as shown in figure. FOFIS, which is located at fire department, uses these data to initialize the forest fire data model from cell-based forestry, meteorological, geographic database, the spatial data model from graphic database, and the resource model from resource database. The forest fire data model is used for providing the geographical, meteorological, and forest information. And, the spatial data model is used for providing the reality that comes from integration of 3-D graphics and aerial photographic images. We can understand configuration of the earth efficiently by using 3-D graphics. Lastly, the resource model is used for generating the forest fire fighting strategy. These three models are handled by the Data Analysis Engine, Graphic Engine, and GA-based Strategy Generation Engine, respectively, for providing various kind of forest fire information. And, the Simulation Engine, which uses the dynamic model, is positioned within the GA-based Strategy Generation Engine for the fitness checking of GA and the prediction of forest fire spread.

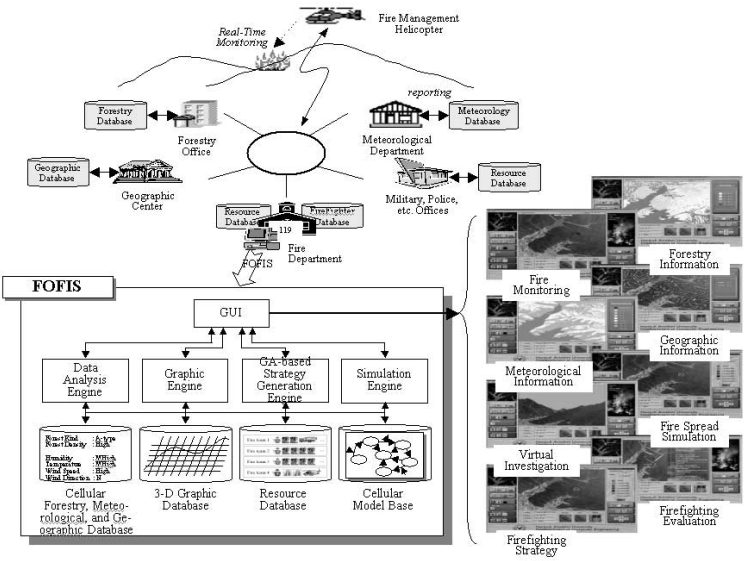



**Fig. 1.** Overview of FOFIS

Through these sorts of database and the works of the engines, the FOFIS provides, not only, the fire monitoring, forestry information, meteorological information and geographical information, but, it also provides the virtual investigation and simulation, through the 3-D graphics, and together with the utilization of fire spreading forecast information, it also provides the fire fighting strategy and the evaluation information on the fire fighting strategy.

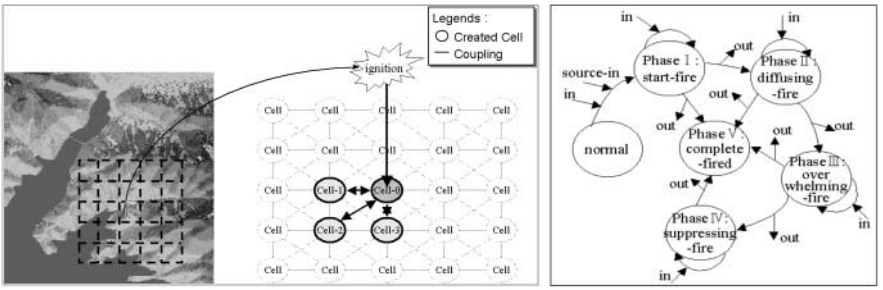
## 3 Cellular Fire Modeling and Simulation Approach

The Discrete Event System Specification (DEVS) formalism provides a means of specifying a mathematical object called a system [8,11]. In the DEVS formalism, basic models are defined by the structure;

$$M =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$$

Where $X$ is the set of external input event types, $S$ is the sequential state set, $Y$ is the set of external event types generated as output, $\delta_{int}$ ($\delta_{ext}$) is the internal(external) transition function dictating state transitions, due to internal(external input) events, $\lambda$ is the output function generating external events at the output, and $ta$ is the time advance function [11].

Multi-component modeling considers models of spatially distributed system with the property of uniformity. A classical prototype of such models is the cellular automata that represent both space and time in discrete form. Discrete




(a) Cellular DEVS model for fore spread

(b) State transition diagram of each cell

**Fig. 2.** Cellular fire modeling

event cellular models preserve the spatial discreteness of cellular automata as well as their space and time invariance. However, the time base is no longer discrete but is continuous, that is to say, there is no intrinsic time step for such models. Events, i.e., cellular state transition, may occur at irregularly spaced intervals, not necessarily synchronized to the beat of a clock as in the cellular automata case. These events are all scheduled to occur as a consequence of the actions of cells; a cell may schedule events to occur by itself as well as to its neighbors, and these events may in turn schedule other events, and so on [12,13].

The simulation model for forecasting the spread of forest fire can be accomplished by using the cellular DEVS model [12]. Fig. 2(a) illustrates the methodology for cellular DEVS modeling. First, the source of forest fire generates the cell-0 model that precedes the forest fire. Then, the cell-0 outputs the forest fire to adjacent cells (here, cell-1, cell-2 and cell-3) according to the state transition caused by time advance of itself (Forest fire is spreading to adjacent area). At this time, the cell-1, cell-2 and cell-3 are generated for the first time and they are coupled with cell-0 so as to organize the new simulation model structure [12,13].

The state transition diagram of each cell model is illustrated in Fig. 2(b). The cell model transforms its state into the new state, in accordance with its own state and fire that is under the mutual interaction with adjacent cells. This is, so to speak, that each cell decides the velocity of forest fire and the quantity of fire spreading to adjacent cells, according to the its own state and the mutual interaction with other cells. The state sets, those cell model has, are a normal state and five phases of fire spread. They are normal (normal state which has no fire), start-fire, diffusing-fire, overwhelming-fire, suppressing-fire, and fired. The variables that influence the state transition of cell model are the velocity of wind, gradient of the inclined plane, humidity, force of fire, temperature, humidity of forest, age of forest and the kind of forest, etc [1,7]. Each variable is divided into High, Middle-High, Middle, Middle-Low, and Low and each has its unique weight

| Simulation Clock | Activated Cell (state) | Graphical view |
|---|---|---|
| 00 : 00 | $C_{5050}$(fire) | |
| 00 : 10 | $C_{5050}$(start-fire) | |
| 00 : 18 | $C_{4949}$(start-fire), $C_{4950}$(start-fire), $C_{4951}$(start-fire), $C_{5049}$(start-fire), $C_{5050}$ (diffusing-fire), $C_{5051}$(start-fire), $C_{5149}$(start-fire), $C_{5150}$(start-fire), $C_{5151}$ (start-fire) | |
| 00 : 20 | $C_{4949}$(start-fire), $C_{4950}$(start-fire), $C_{4951}$(start-fire), $C_{5049}$(start-fire), $C_{5050}$ (diffusing-fire), $C_{5051}$(start-fire), $C_{5148}$(start-fire), $C_{5149}$(diffusing-fire), $C_{5150}$ (start-fire), $C_{5151}$ (start-fire), $C_{5248}$(start-fire), $C_{5249}$(start-fire) | |
| ... | ... | ... |
| 01 : 00 | $C_{4546}$(start-fire),$C_{4547}$(start-fire),$C_{4548}$(start-fire), $C_{4549}$ (start-fire), $C_{4550}$(start-fire), $C_{4551}$(start-fire), $C_{4646}$(start-fire), $C_{4647}$ (diffusing-fire), $C_{4648}$ (diffusing-fire), $C_{4649}$ (diffusing-fire), $C_{4650}$(diffusing-fire), $C_{4651}$(start-fire), $C_{4745}$(start-fire), $C_{4746}$ (diffusing-fire), $C_{4747}$(overwhelming-fire), $C_{4748}$(overwhelming-fire), $C_{4749}$(overwhelming-fire), ..., $C_{5444}$(start-fire), $C_{5453}$(start-fire), $C_{5544}$(start-fire), $C_{5545}$ (start-fire), $C_{5546}$(diffusing-fire), $C_{5547}$(diffusing-fire), $C_{5548}$(diffusing-fire),$C_{5549}$(start-fire),$C_{5550}$(diffusing-fire),$C_{5551}$(start-fire), $C_{5552}$(start-fire), $C_{5553}$ (start-fire), $C_{5646}$(start-fire), $C_{5647}$(start-fire), $C_{5648}$(start-fire), $C_{5649}$(start-fire), $C_{5650}$(start-fire), $C_{5651}$(start-fire) | |
| ... | ... | ... |

Legend : : start-fire : diffusing-fire : overwhelming-fire : suppressing-fire : complete-fired

**Fig. 3.** Simulation trajectory of forest fire spreading

value according to the power of influence for forest fire spreading. The velocity of forest fire spreading and the quantity of fire spreading to adjacent cells are decided by means of the result from "(the level of wind velocity x weight1) + (the level of gradient x weight2) + ... + (the level of tree kind x weight n)" as well as the wind direction and the inclined plane [4]. These equations need to be improved in conformity with the exact experiment.

Fig. 3 partially shows the result of a forest fire simulation test in which the strong wind with northeast direction for initial condition is assumed. As shown in Fig. 3, initial fire begins at point cell5050, and, as time passes, one is able to know that it is spreading into the neighboring cells, as the state transition is realized. For an example, at simulation clock 00:00, initial fire starts at cell5050, and after 10 minutes, the state of cell5050 changes to start-fire, and the full-scale forest fire begins to spread. Moreover, after elapsing 8 minutes, cell4950, cell4951 and other cells, which are in the vicinity of cell5050, have been affected by cell5050, and have changed their states into the start-fire states, and the state of cell5050 changes to overwhelming state passing through the diffusing-fire state.

## 4 Automated Methodology for Generating Fire Fighting Strategy

In order to generate the efficient forest fire fighting strategy, we have proposed the automated methodology (see Fig. 4) that uses the Rule-based System Entity Structure (RUSES) [8,9] and simulation-based genetic algorithm (GA) [10] as PHASE I and II respectively. PHASE I composes the SES, which expresses all

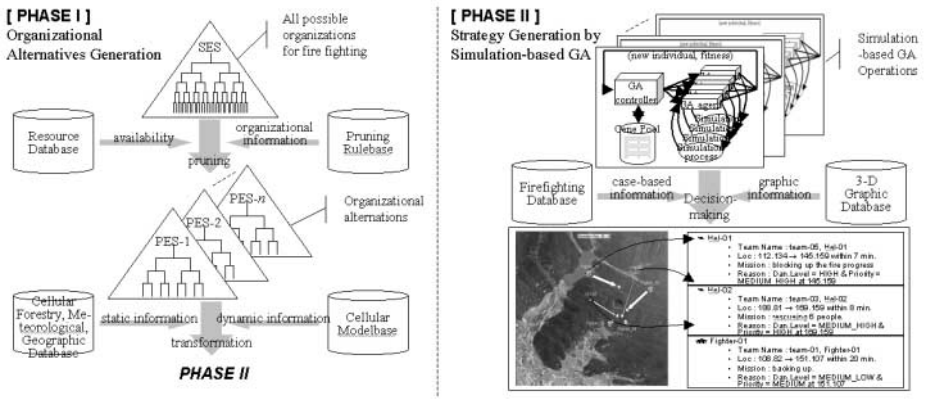



**Fig. 4.** Fire fighting strategy generation methodology

the resources that are possible to be applied in the fire fighting, hence, it is to generate the organizational alternatives of the fire brigade such as human and equipment using the available fire-fighting resource information and the associated selection/synthesis rules of the RUSES. Consequently, the pruned fire-fighting organization, that is formed, can be directly applied to the cellular fire model structure discussed earlier so that the simulation is proceeded by generating the fire spread behavior with respect to the extinguishments behavior resulting in a certain fitness value. Based on the GA approach with this fitness value, the simulation is repeated until the proper fitness level is achieved by creating new population with chromosome composed with the work-area, work-kind, and work-duration of each resource.

### 4.1 PHASE I: Organizational Alternatives Generation by the Rule-Based SES

The SES is a representation scheme that contains the decomposition, coupling, and taxonomy information of a system [8,11]. The SES contains three types of nodes - entities, aspect, and specialization - which represent three types of knowledge about systems. The entity node, which may have several aspects and/or specializations, corresponds to a model component that represents a real world object. One application of the SES framework relates to the design of a system. Here the SES serves as a compact knowledge representation scheme of organizing and generating the possible configurations of a system to be designed. To generate a candidate design, we can use a pruning process that reduces the SES to a PES (Pruned Entity Structure).

The pruning process restricts the space of possibilities for selection of components and coupling that can be used to realize the system being designed. Thus we can assume that design may now be reduced to the synthesis problem. Synthesis involves putting together a system from a known and fixed set

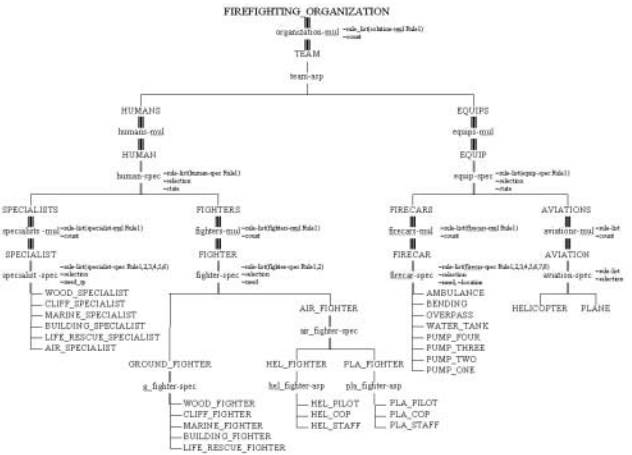



**Fig. 5.** SES for resource allocation

of components in a fairly well prescribed manner. In the synthesis problem, we are modeling a rather restricted design process, one amenable to automation by extracting concepts and procedures from expert's knowledge and experience, augmenting them and molding them into a coherent set of rules. To do this, we have developed the rule-based SES by integrating the rule-based expert system methodology with conventional SES [9].

At any point in a SES, where more than one choice is presented, e.g., a selection between two different special types of a component, attributes and rules are attached to the components to guide in the pruning process. A pruning procedure checks these attributes and associated rules to select the appropriate entities. The selected entities are used to construct design models. The pruning algorithms generate all design model structures that conform to the design objectives and constraints.

To deal with structural constraints imposed on the system being designed, the design model development process is augmented with synthesis rule specializations. The synthesis problem is conceived as a search through the set of all pruned entity structures. Based on these concepts, a canonical production rule scheme for a hierarchical design model synthesis called RUSES (Rule-based System Entity Structure) has been developed [10]. Thus, we have been employed the RUSES for allocating resources, such as equipment and human power, so as to organize the appropriate teams for forest fire fighting.

The entire SES for the resource allocation is presented in Fig. 5.

This structure includes all possible resources for constructing the forest firefighting teams. The FIREFIGHTING-ORGANIZATION, which is the root entity, is formed with many TEAM. And TEAM is composed of HUMANS and EQUIPS entity. The HUMAN entity, furthermore, is specialized into the SPECIALISTS and the FIGHTERS. Similarly, the EQUIP entity is also further

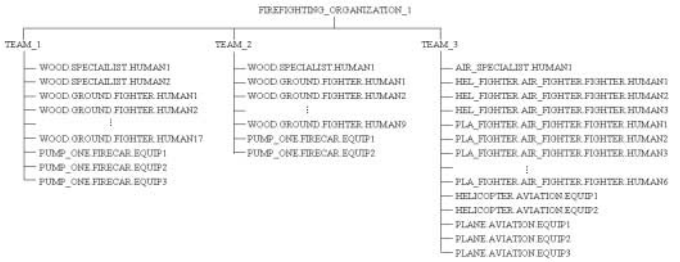



**Fig. 6.** Example of PES

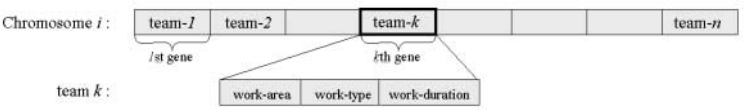



**Fig. 7.** The formation of chromosome and gene

specialized into the FIRECARS and AVIATIONS entity. The entities that have the decomposition node have the composition rules, and the entities that have the classified node have the selection rules, so if this structure has an appropriate attribute, then the optimal structure can be organized.

An example, after applying the RUSES, is illustrated in Fig. 6. Fig. 6 is the pruned entity structure (PES), which is the result of pruning and cutting on SES with constraints, such as, the case that the fuel type of area is wet wood and the size of forest fire is small.

### 4.2 PHASE II: Strategy Generation by the Simulation-Based GA

Genetic algorithms have its base on the principle of biological evolution for a probability search, learning or optimization. The basic four steps of genetic algorithms are the generation of chromosome, fitness checking, selection and genetic operations [10].

**Generation:** This is the step at which chromosomes are formed with the work-area, work-type, and the work-duration of the fire fighting brigade as the gene. As in Fig. 7, chromosome that is a gathering of genes is made of n number of genes; the n represents the number of team which are formed through the RUSES in PHASE I. For an example, the kth gene has the information relating with the work of kth team and are formed with three fields - work-area, work-type, and the work-duration. The respective chromosomes are formed randomly at first and, by applying the heuristic knowledge, it is possible to effectively shorten the overhead of the computation time.

**Table 1.** Example: generated fire fighting strategy

| team-name | arrival-time (day, hour, min) | work-area (x,y-coord) | work-type | work-duration (hour) |
|---|---|---|---|---|
| Team-1 | 1, 12-45 | (14, 94) | cutting-tree | 1.5 |
| Team-2 | 1, 15-00 | (13, 90) | fire-walling | 0.5 |
| Team-3 | 1, 15-30 | (15, 100) | water-pumping | 3.0 |
| Team-4 | 1, 15-45 | (15, 101) | water-pumping | 2.5 |
| ... | ... | ... | ... | ... |

**Fitness Checking:** This is the step in estimating the fitness of the chromosome. As in Fig. 4, the genetic algorithm controller distributes the chromosomes of one generation to the respective GA agent and GA agents perform the simulation of forest fire spreading according to the work-area, work-type, and the work-duration. For example, if work-type is water pumping, then humidity of work-area decreases; based on that result, the fitness checking is performed. At this point, the fitness value is determined by the results of the simulation values, the time required for extinguishing the forest fire, and the amount of the area of the damaged zone.

**Selection:** This is the step in which the chromosomes are composed to be existed in the next generation, depending on the fitness value. According to the fitness rank, the higher rank of chromosomes exists in the next generation without any variation, and the lower rank of chromosomes is weeded out.

**Genetic Operations:** This is the step in which the genetic operation of the crossover, mutation, and etc., are applied to the chromosomes to be in existence in the next generation. In this research, the crossover operation is applied on 60% of chromosomes. The mutation operation used here is applied on 3% of chromosomes. Through these genetic operations, chromosomes that mean firefighting strategy are reproduced for the next generation. And, they are going under the same processes, such as fitness check, selection and genetic operations. After several iterations through generations, most optimal chromosomes, so to speak, the optimal fire-fighting strategies can be generated.

An example of the result, after going through the genetic algorithm as described above, is as shown in Table 1. For instance, Team-3 should be arrived at location 15(x-coordinate), 100(y-coordinate) at 15:30 on the appointed day, and their duty is water pumping, and work duration is scheduled up to 3 hours.

## 5 Case Study: FOFIS

As a case study of this research, we have been tested the virtual forest fire scenario at Chung-do area in Korea as shown in Fig. 8. In the figure, the user

**Fig. 8.** GUI of FOFIS

interface of FOFIS is largely divided into five fields. The left side is the control button section for the information indicators of the FOFIS system, simulation control, the result output and etc.; the middle is the output section to indicate the simulation result and the monitoring result; the one below is the dynamic information output section which is to indicate the dynamic information of the weather conditions, wind direction and etc.; to the right of the upper side is the static information output section which is to indicate the static information, such as, the topography, the height of the sea level, and etc.; lastly, the right side of below is the output control section which is to control the zoom-in/out of the output section and the 3-D output.

Initially, when there is an outbreak of a forest fire, then immediately, the information on the site of the initial forest fire, its size, and etc., which are sensed by the sensor of the forest guard personnel, aerial surveillance equipment, and etc. is inputted into the Forestry Office and to other related organization. This data, through the network, is delivered to the FOFIS system, which is controlled by the Head Office of the Fire Stations, activates the FOFIS. From this point on, the FOFIS system operates as the simulator and as the fire fighting strategy generator. Out of the buttons on the left, the fire fighting information button indicates the information on the fire extinguishing resources situation, which are currently being utilized. Through the fire prevention information button, the genetic algorithm process initiates the formation of the fire fighting strategy.

For the formation of the fire fighting team, with the utilization of the available fire fighting resources, the FOFIS forms the SES by utilizing the available fire fighting support that is stored in the Resource Database (refer to Fig. 1), and, by utilizing the RUSES, the PES, namely, the fire fighting resources to be utilized, is formed from SES. Since the initial condition of the size of the forest fire was assumed to be not large in this test, the resource for 12 fire brigades altogether were assigned. Fig. 9 shows the structure of PES that is formed by the pruning process of RUSES, and the fired rule list by the pruning process.

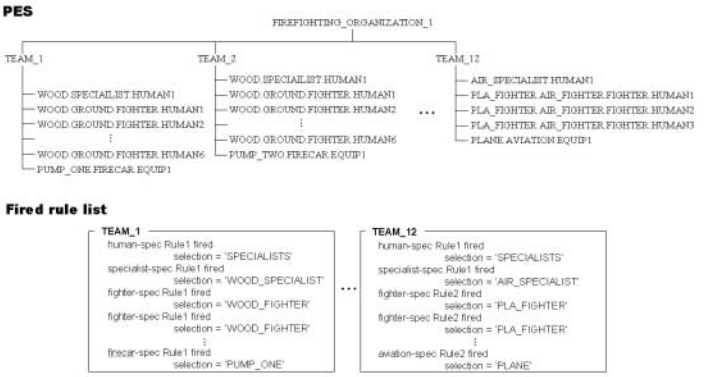


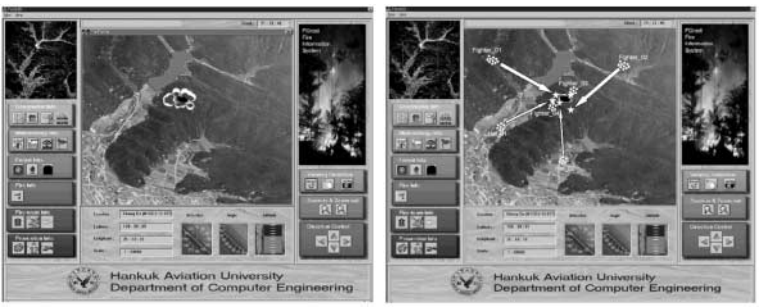

**Fig. 9.** PES and rule list by RUSES

Continuously, the genetic algorithm is performed for the formation of the best fire fighting strategy at PHASE II. Initially, the work-area, the work-type, and the work-duration of each team are formed, and the fitness checking is performed by inspecting this result through simulation. The simulation process of FOFIS illustrates the time-based diffusion of forest fire at a time interval of 1, 2, and 3 hour (see Fig. 8)

And, Fig. 10(a) shows the procedure of GA that adapts strategies to the optimal one. In this Figure, the transition of work-area is illustrated. In the case of initial generation, the work-area of each fire brigade is spread around with no relation of the firing spot, but the work-area is concentrating toward the firing spot after several generations. At last, we can get the optimal team organization and forest fire fighting strategy. This result can be displayed graphically or in texture as shown in Fig. 10(b). The figure shows the effective movement on the current fire brigade's (both of ground and aerial) position based on the best fire fighting strategy formed.

## 6 Conclusions

The objective of this paper was to design and develop the efficient decision support system for forest managers that would help them in their decision-making tasks during the forest fire. To do this, we have adopted the advanced simulation techniques as well as the genetic algorithm for generating the forest fire fighting strategy. Furthermore, in order to support effective decision making, GIS database utilizing the 3-D graphic was introduced, and, it has suggested the formation of the available fire fighting resources through the SES and pruning techniques, the fire model through the utilization of the geographical, meteorological, and forest information, and the simulation process utilizing the cellular modeling techniques. Additionally, it has successfully realized the FOFIS system, which is the source of the decision making support system, utilizing the




(a) Evolution of work-area of each team (500th generation)

(b) Generated fire fighting strategy

**Fig. 10.** Strategy generation by simulation based GA

proposed methodology, and, it has shown the effective coping with the simulation forest fire situation at Chung-do, Kyungsang Buk-do area in Korea. As the future works, we need to research forest fire related elements continuously and to validate the proposed system through test on a real situation.

## References

[1] A. M. Grishin: General Mathematical Model for Forest Fires and Its Applications. Combustion, Explosion, and Shock Waves, Vol. 32, No.5. (1996) 503–519
[2] J. R. Coleman, A. L. Sullivan: A real-time computer application for the prediction of fire spread across the Australian Landscape. SIMULATION J. (1996) 230–240
[3] Green, D. G., Tridgell, A. and Gill, A. M.: Interactive simulation of bushfires in heterogeneous fuels. Mathematical and Computer Modelling, Vol. 13, No. 12. (1990) 57–66
[4] F. Ricci, A. Perini, and P. Avesani: Building first intervention plans: the forest fire case. Proc. of Artificial Intelligence Research in Environmental Science, Biloxi, Mississipi (1994)
[5] F. Ricci, A. Perini, and P. Avesani: Combining CBR and Constraint Reasoning in Planning Forest Fire Fighting. Proc. of 1st European Workshop on Case-Based Reasoning, Kaiserslautern (1993)
[6] A. Perini and F. Ricci: Constraint Reasoning and Interactive Planning. Workshop on Constraint Languages-Systems and their use in Problem Modeling, N. Y. (1994)
[7] J. L. Wybo: FMIS: A Decision Support System for Forest Fire Prevention and Fighting. IEEE Trans. on Eng. Management, Vol. 45, No. 2, (1998) 127–131
[8] B. P. Zeigler: Object-Oriented Simulation with Hierarchical, Modular Models. Academic Press (1990)
[9] S. D. Chi, J. S. Lee, J. K. Lee and J. H. Hwang: NETE: Campus Network Design Tool. Proc on IASTED (1997)

[10] D. E. Goldberg: Genetic Algorithms in search, optimization, and machine learning. Addison-Wesley (1989)
[11] B. P. Zeigler: Multifacetted Modeling and Discrete Event Simulation. Academic Press (1984)
[12] T. H. Cho and S. D. Chi: OName-directed coupling applied to cellular model: river pollution example, Proc.on MODSIM 95, Newcastle, Australia (1995)

# Knowledge Maintenance and Sharing in the KM Context: The Case of P–Truck

Stefania Bandini, Sara Manzoni, and Fabio Sartori

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo)
University of Milan - Bicocca
via Bicocca degli Arcimboldi, 8
20126 - Milan (Italy)
tel +39 02 64487857 - fax +39 02 64487839
{bandini,manzoni,fabio.sartori}@disco.unimib.it

**Abstract.** This paper illustrates a Knowledge Management framework developed in the context of the P–Truck Project, that aims to support the design and manufacturing of innovative products in a restricted and specific domain (i.e. the design and manufacturing of truck tires) through a Knowledge Based System approach. The domain is characterized by heterogenous knowledge, that can be difficulty captured with generic knowledge engineering methodologies and tools. Thus, a dedicated Knowledge Elicitation tool (KEPT) has been designed and implemented into the P–Truck system for this task. The main feature of KEPT is the possibility to manage the heterogeneous knowledge concerning the different phases of the production process in a *centralized* fashion, with benefits from the knowledge maintenance and sharing standpoints. Moreover, KEPT allows to support the different types of knowledge–based systems (i.e. rule–based and case–based) that within the P–Truck system support domain experts in handling tire design and production.

## 1 Introduction

Knowledge Management (KM) can be considered as the process through which organizations generate value from their intellectual and knowledge–based assets [17, 10]. Most often, generating value from such assets involves sharing them among employees, departments and even with other companies in an effort to devise best practices, that are the most important resource for an enterprize to be competitive on global markets. Thus, enterprizes' interest in supporting KM through the adoption of computer–based tools and methodologies has significantly grown over last years, becoming a trend within the different communities of researchers in Computer Science. According to [19], such tools and methodologies can be generally led to the Artificial Intelligence (AI) area, e.g. Knowledge Based Systems (KBS), data mining, e–learning application tools and so on. Thus, a lot of researchers in AI have focused their attention on the development of sophisticated tools and methodologies to deal with the KM area and to provide companies with generic methods and tools to capitalize their knowledge. The

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 499–510, 2003.
© Springer-Verlag Berlin Heidelberg 2003

results of these efforts seem to be encouraging: many KM frameworks have been recently designed and (partially) implemented to this aim, and someone is trying to produce standards [18]. Anyway, it is not simple to evaluate the effectiveness of such generic methodologies, since they are still at an initial state. Moreover, KM systems are often devoted to solve very specific problems related to very specific domains, so sometimes dedicated tools and methods could be preferable with respect to generic ones.

In particular, although a lot of Knowledge Based Systems have been developed for dealing with several domains [12], the phase of knowledge acquisition and representation is still the main problem of this type of tools [8]. Knowledge engineering methodologies, such as CommonKADS [2] and MIKE [3], have been proposed as standard and generalized solutions to solve this problem: anyway, the knowledge acquisition and representation tasks can often be tackled more precisely with specific tools, due to the specific nature of involved knowledge, that can not always be captured exploiting methodologies designed for heterogeneous domains [5].

This paper illustrates a specific approach to Knowledge Management, based on KBS technology, developed in the context of the P–Truck Project. The University of Milan–Bicocca and the Truck Business Unit of Pirelli Tires are collaborating to develop an integrated KM system to support the design and manufacturing of truck tires. The P–Truck system is made up of different KBSs, each one dedicated to a specific tire production phase. In particular, knowledge acquisition sessions with the experts of Pirelli have pointed out that the different phases of the truck tire life–cycle are accomplished by dedicated Communities of Practice [20] (CoPs), that sometimes interact to solve complex problems. Thus, the domain of P–Truck is characterized by heterogenous knowledge, that may require frequent updates and that must be shared among the involved CoPs. In order to take care of the complex nature of the involved knowledge, a domain–specific Knowledge Elicitation framework (i.e. KEPT, Knowledge Elicitation module of P–Truck) has been designed and implemented. The development of KEPT has allowed to overcome the main problems that arises when a KBS approach is undertaken: maintenance of the knowledge base and knowledge sharing among experts and integrated KBSs. KEPT is described with further details in paragraph 4, after a brief introduction to the truck tire production domain and a description of the general architecture of P–Truck. The paper ends with some conclusions and future works.

## 2 The P–Truck Domain

The P–Truck Project aims to support the decision making process of experts involved in the truck tire production at the Business Unit Truck of Pirelli Tires.

A truck tire is a very complex product: according to [9], it can be considered as a *chemical device*, that is made up of both chemical components and other elements. In particular, a truck tire is composed of rubber compounds (i.e. the chemical part), that are responsible for all the thermal–mechanical char-

acteristics of the tire, and metallic reinforcements that provide the tire with the necessary rigidity. Different from chemical commodities, whose life–cycle is focused on the optimization of manufacturing process, the one of a chemical device is centered on the product innovation, in order to meet the requirements of evolving markets they are devoted to. In the case of truck tires, it is necessary to optimize a lot of performance (e.g. tensile strength, resistance to fatigue), and the importance of them varying according to the kind of market they will be sold on (e.g. South America, Europe, Asia).

The life–cycle of a truck tire can be divided into the following main phases:

- Design of rubber compounds: a rubber compound is a blend of different ingredients, both natural (e.g. natural rubber, resins) and synthetic (e.g. carbon blacks, oils). This design phase has to decide the blend composition, identifying a set of ingredients and their amount, in order to achieve the performances that are required for the blend and for the tire (e.g. tensile strength, resistance to fatigue).
- Mixing: the ingredients must be suitably mixed in order to obtain a homogeneous blend with the required viscosity (again, related to rubber compound and tire properties).
- Semi–manufactured production: reinforcements are added to rubber compounds, producing the different parts that will compose the tire.
- Assembly: semi–manufactured parts are assembled into a semi–finished product (i.e. green–tire, in the tire jargon).
- Vulcanization: the green tire is processed in order to give it the required thermal–mechanical features.

The above summarized life–cycle is sufficiently general to characterize all tire production realities, where different specific way to perform it can be possible. In the specific case of the Business Unit Truck of Pirelli Tires, a knowledge acquisition campaign has revealed that each of the above summarized phases is accomplished by a specific Community of Practice (CoP). The aim of the P–Truck Project is to support the CoPs that are involved in the Design of rubber compounds (i.e. compound designers), Mixing (i.e. Mixing technologists) and Vulcanization (i.e. Vulcanization technologists).

## 3 The P–Truck Architecture: An Overview

The architecture of P–Truck has been designed in order to have a *centralized knowledge repository* and a *distributed problem solving strategy*. As will be better explained later, the designed architecture provides a lot of benefits, in particular, from the knowledge maintenance and sharing viewpoints.

Figure 1 shows a draft architecture of the P–Truck system, that can be divided into two main parts: a first part is devoted to capture, represent and store expert knowledge (i.e. KEPT, Knowledge Elicitation module of P–Truck), and another one is devoted to process this knowledge (i.e. KPM, Knowledge Processing Modules). In particular, the KPM is made up of four Knowledge Based Systems:

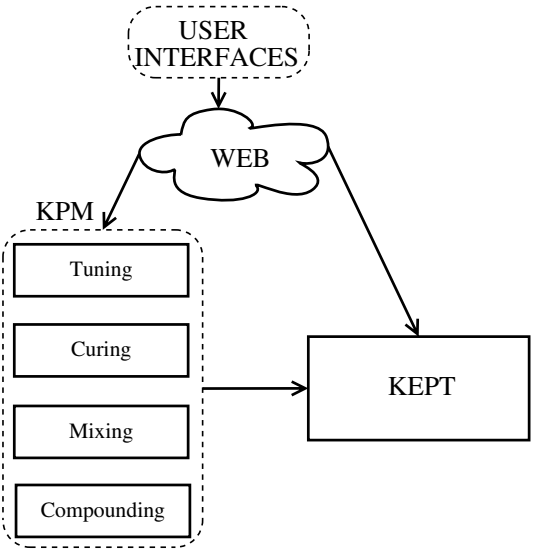



**Fig. 1.** P–Truck Modules

- Compounding, a rule–based system that supports the CoP of compound designers;
- Mixing, a rule–based system that supports the CoP of Mixing technologists;
- Curing, a case–based system that supports the CoP of Vulcanization technologists;
- Tuning, a case–based system that support a CoP whose aim is to handle possible anomalies that may occur during the production phases. Members of this CoP come from other design CoPs and, when needed, they negotiate to solve the encountered anomaly.

From the knowledge maintenance point of view, the development of KEPT has allowed to provide experts with a knowledge repository and a flexible tool to manage it (i.e. visualize, update and retrieve). It is evident that a single knowledge base that collects data and knowledge coming from different sources is preferable from this point of view. Although other approaches, like for instance Agent–Based technology, have been proposed for complex system design [13] and integration [11], the solution adopted in P–Truck allows to avoid problems that must be considered when dealing with distributed knowledge bases (e.g. data inconsistency, no functional issues and so on). A centralized knowledge base facilitates all activities that require to access it and, at the same time, it is an advantage also from the knowledge sharing point of view. In fact, this solution allows to provide different KPM components with a specific view on the knowledge base that they can access to perform their tasks. Finally, new KBSs that exploit the P–Truck knowledge base content can be simply added

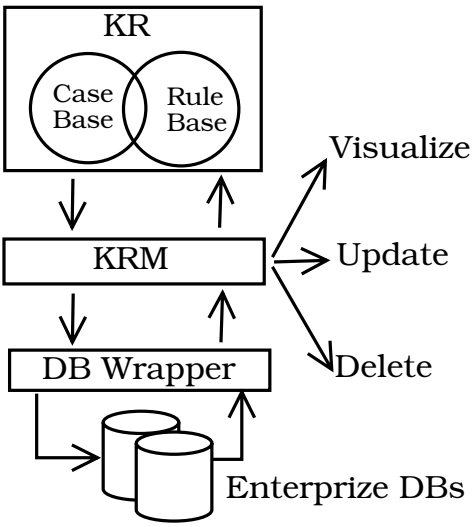



**Fig. 2.** The architecture of KEPT

and provided with an appropriate view on the knowledge base content (see in Section 4.3 the case of the Tuning module).

Another important feature of the KEPT knowledge repository concerns its internal partition into two main parts. Each partition serves a different type of knowledge–based KPM. In particular, a first partition represents knowledge that is exploited by rule–based modules, and the other one constitutes the case memory of case–based modules. Within the P–Truck Project particular emphasis has been dedicated to the definition of the most suitable KBS approaches to support the different CoPs involved in the tire production process. As previously introduced, a rule–based approach has been adopted for dealing with the decision making process of compound designers and mixing technologists, while the tuning of production process and the support of vulcanization technologists have been managed through a Case Based Reasoning approach [1]. Different choices have been motivated by the different nature of decision making processes that had to be supported by the KPM modules. At one hand, the design of rubber compounds is based on well–known chemical relationships between ingredients, blend properties and tire performances. Those relationships can be adequately captured, represented and stored into a rule base. Analogously, the design of mixing processes follows well–known and experienced rules that bind compound ingredients and machinery features. On the other hand, there are no explicit relationships between production anomalies and their solutions. Production experts, due to scheduling and costs requirements, often reason by analogy, and apply to anomalous production situations solutions that have already been experienced in the past.

## 4 KEPT: The Knowledge Elicitation Module of P–Truck

In this section KEPT will be further described to explain how it allows the acquisition and representation of knowledge belonging to the CoPs supported by the P–Truck Project. Moreover, it will be shown how KEPT allows different KPMs to exploit this knowledge to support those CoPs. As shown in Figure 2, KEPT is composed by two main parts: a *Knowledge Repository* (KR) that allows the knowledge representation and storing, and a *Knowledge Repository Manager* (KRM), that provides users and KPM modules with all the functionalities to manage the KR content (i.e. visualization, updating, deletion and so on).

### 4.1 Knowledge Acquisition and Representation

KR is the storage mean for the knowledge owned by the CoPs supported by P–Truck. As previously introduced and represented in Figure 2, KR is divided into two logical parts: a first one devoted to represent knowledge supporting KPM rule–base modules, and a second one to represent the case structure KPM case–based modules require. KRM allows users (i.e. P–Truck users and the KPM modules) to manage the KR content, providing them with functionalities like visualization, deletion, retrieval and update of the stored knowledge. Figure 3 shows a screenshot of the graphical interface of the P–Truck module that allows users to visualize the KR. This graphical representation reproduces the internal structure of the KR that users can navigate to consult the KR content (see the following sections for more details on KR structure).

The DB Wrapper module acts as a configurable interface between KRM and enterprize databases that contain the information about rubber compound ingredients, ingredient properties, machinery features, production factories, and so on. The DB Wrapper has been developed in order to provide the KR with possibly heterogeneous and geographically distributed data that are already available in the enterprize information system. Wrapped enterprize data are organized and structured within the KR according to expert knowledge and integrated with other useful data. For instance information about ingredient properties are structured in a way that takes into account their relationship to blend and tire features. The DB Wrapper has been designed and developed in such a way that allows it to be independent from the nature of information sources: when a KPM module (or a P–Truck user) needs some data, its request is sent to the DB Wrapper. The latter translates the query into a suitable format for the information source that contains required data (e.g. an Oracle Database), executes the query and returns the result into a suitable format for the requiring KPM (or the user interface).

A very important benefit obtained by the adoption of such architecture for KEPT is the easy maintenance of the KR knowledge: the implemented KRM has provided experts with a user–friendly tool to visualize and update KR content whenever they wish. Moreover, the KR content is *transparent* to KPM modules: this means that KPM modules do not need to be modified when changes in the KR occur. The reason for this is that all the tacit knowledge acquired from CoPs'

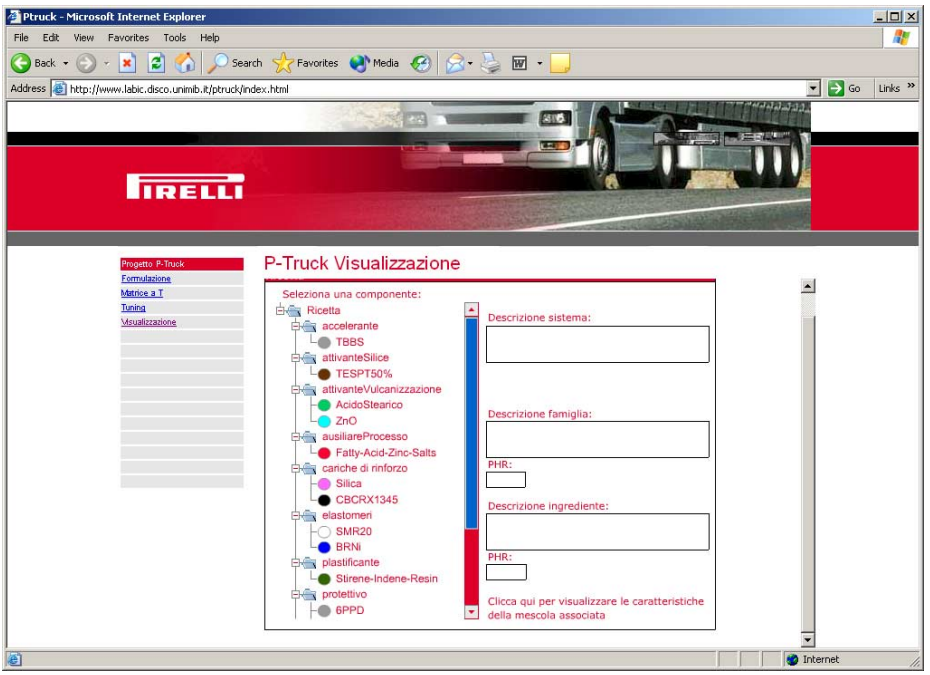



**Fig. 3.** An example of rubber compound components

members has been modelled in such a way that a static part has been exploited to define KPMs and the KR structure according to which dynamic data that may need to be updated are organized. In this way, KPM modules uses the KR knowledge as simple structured data and thus interpret it to perform their tasks.

Also the sharing of knowledge among experts is improved by such choices: since the different CoPs interact in order to solve complex problems, some views of the knowledge base could be composed of heterogeneous knowledge representing different competencies. Thus, in order to solve new problems, it is sufficient to add new modules to KPM component of P–Truck, working on a view of the knowledge base containing all the knowledge (and possible quantitative information coming from enterprize's archives) necessary to tackle the situation. In the P–Truck system, sharing knowledge among CoPs means creating a suitable view of the knowledge base that takes care of knowledge owned by each of them.

In the following, two examples of how expert knowledge has been modelled and structured in the KR and KRM are described: in particular, knowledge models built to support the *Compounding* and the *Tuning* KPM modules are illustrated.

**Table 1.** An example of T–Matrix

| | | | | |
|---|---|---|---|---|
| TP 1 | ⊙ ↑ | ○ ↑ | ⊙ ↓ | ⊠ |
| TP 2 | ⊠ | ⊙ ↑ | ▲ ↓ | ○ ↑ |
| TP 3 | ○ ↑ | ⊠ | ▲ ↓ | ⊙ ↓ |
| TP 4 | ⊙ ↓ | ▲ ↑ | ⊠ | ⊙ ↑ |
| TP 5 | ⊠ | ⊙ ↓ | ○ ↑ | ⊠ |
| | BF 1 | BF 2 | BF 3 | BF 4 |
| RI 1 | ⊙ ↓ | ▲ ↓ | ⊠ | ⊠ |
| RI 2 | ○ ↑ | ⊙ ↓ | ▲ ↑ | ⊙ ↑ |
| RI 3 | ⊙ ↓ | ⊠ | ⊠ | ▲ ↑ |
| RI 4 | ▲ ↓ | ⊙ ↑ | ▲ ↑ | ⊠ |

### 4.2 KEPT Support to Rule–Based Modules

As an example of how KEPT allows a simple management of knowledge coming from CoPs involved in truck tire production the acquisition and representation of compound designers' expertise is considered: a model of the involved knowledge has been realized, that has allowed to provide compound designers with a uniform representation of their core knowledge. This model is an extension of the *Abstract Compound Machine* (ACM) described in [4].

The fundamental entities considered in designing the knowledge model are *tire*, *recipe* and *ingredient*. Tires are final products of the manufacturing process, and must satisfy required *performances* (TP) (e.g. low cost, employee, and so on). The analyzed CoP designs tire rubber compounds, defining for each of them a recipe, that is a list of ingredients with related quantities. A recipe is made or modified according to a list of *features* (RF) the related rubber compound have to meet (e.g. tensile strength, resistance to fatigue, aging, and so on)

Product innovation is achieved through a sequence of *interventions* on recipes (RI) previously used in tire production. Some examples of interventions are the augmentation of the amount of an ingredient present in the recipe or the substitution of an ingredient with another one, belonging to the same family. Interventions are chosen in order to modify blend features and, consequently, tire performances according to marketing requests.

The very important knowledge related to product innovation for truck tire stands in two relationships, called *Compounding Relation* and *Design Relation*. The first relationship binds interventions aiming to modify a recipe and RFs, while the latter describes the correlation existing between RFs and TPs.

Table 1 reports an example of how compounding and design relations have allowed to structure and manage the KR content referring to compound design activity. Upper parts of the table are referred to as *Design Relation* while *Compounding Relation* is indicated in the lower rows of the table. Symbols used for describing correlations and proportionality and the related meanings are reported in Table 2.

**Table 2.** Design and Compounding relationship vocabulary

| | | |
|---|---|---|
| Correlation | ⊙ | Strong |
| | ○ | Good |
| | ▲ | Weak |
| | ⊠ | No corr. |
| Proportionality | ↑ | Direct |
| | ↓ | Inverse |

The vocabulary represented in Table 2 allows the Compounding KPM to interpret these relationships and support compound designers in their tasks.

### 4.3 KEPT Support to Case–Based Modules

Case Based Reasoning (CBR [1]) is a problem solving paradigm aiming to find solutions to complex problems by analogy with similar past situations, suitable to deal with domains which have not been fully understood and modelled. One of the most important issues to be tackled in the application of the CBR approach is the definition of case structure [15]. This structure may be fixed or variable [7], depending on the complexity of the knowledge involved: the choice of significant attributes strongly influences the building of a good similarity function among cases, that is fundamental in order to effectively retrieve past cases similar to the current one.

Within the P–Truck project the CBR paradigm has been applied to design and develop the Tuning and Curing modules: in particular, this section focuses on the first, whose aim is to support the KM of people dedicated to handle unpredictable anomalies during the production process (like the lack of one or more blend ingredients during the mixing phase, problems due to machinery maintenance and so on) [6].

The organization of the P–Truck knowledge repository enhances knowledge sharing in this framework. To handle production anomalies is a decision making process that requires a lot of competencies owned by different CoPs involved in the whole production process. To solve production anomalies, often it is not sufficient exploiting the experience and knowledge of a single CoP. Sometimes, in fact, an interaction among different CoPs is necessary. For example, suppose that an ingredient that compound designers designed to be part of the blend recipe is not available at mixing time. In order to take care of production scheduling and costs, mixing technologists have to ask compound designers if the lacking element could be substituted by another one preserving the features of the designed rubber compound. Compound designers may produce a new blend recipe substituting the missing ingredient with another one, and ask mixing technologists to evaluate if the proposed blend can be correctly processed by mixing machineries. Thus, the CoPs of compound designers and mixing technologists interact (i.e. *negotiate*, in the CoP jargon [20]) to provide a new suitable blend recipe that

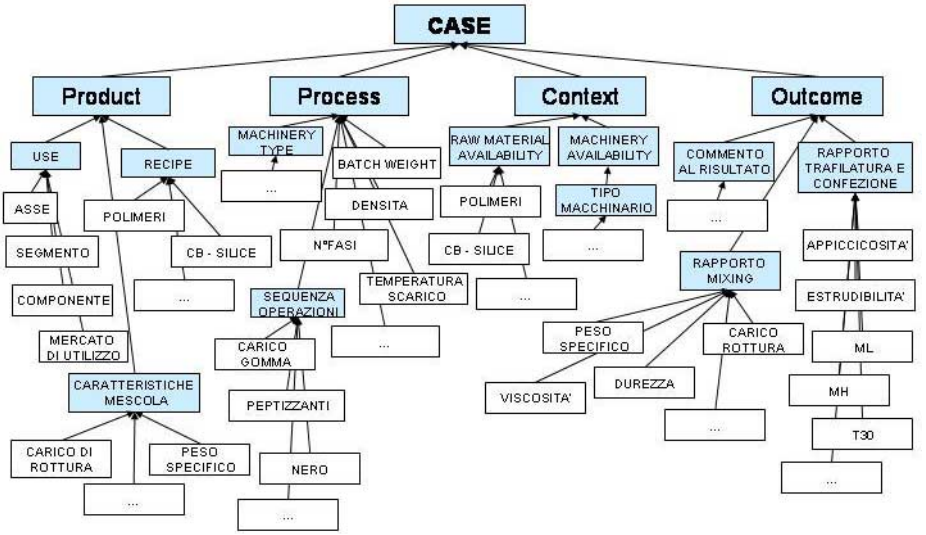



**Fig. 4.** The representation of a portion of the tree structured cases implemented in the P–Truck KEPT module

can be processed as the original one. In other words, they negotiate in order to overcome the encountered production anomaly (i.e. the lack of an ingredient) without originating new ones (i.e. difficulties in processing the ingredients).

The centralized knowledge base of the P–Truck system allows the Tuning system, which support the involved CoP in this negotiation process, to exploit the KR content referring to mixing and compound design. When a production anomaly has to be tackled, a suitable view of the KR is provided to the KRM and a CBR retrieval process is performed by the Tuning KPM.

Figure 4 shows a portion of the adopted case structure: it is possible to note how it contains information concerning both the compound designer task (i.e. the *Product sub–tree*) and mixing–vulcanization technologists (i.e. the *Process sub–tree*). The *Context sub–tree* describes productive contexts (e.g. plants and machineries) and possible anomalies occurring during the manufacturing process (e.g. lack of raw materials or machineries). This tree–like structure has been integrated into KEPT (see Figure 2) in order to suitably allow an incremental enrichment of case structure when new information is available and has to be integrated. Adding new information basically means providing the tree structure with a new node, that can be attached to the root (e.g. if the node represents a new type of information) or to an internal node (e.g. if the node represents a specialization of previously inserted information). Interested readers can refer to [16] for more details about the motivations, features and advantages of the tree–like structure adopted for P–Truck case base.

## 5 Conclusions and Future Works

The paper has presented KEPT, the Knowledge Elicitation module of P–Truck. The aim of the P–Truck project is to build a specific–domain framework based on KBS technology for supporting experts of Pirelli Tires in their decisional process about the design and manufacturing of truck tires. This domain is characterized by heterogeneous knowledge, that can not always be acquired and represented exploiting generic methodologies: so a dedicated knowledge elicitation tool (i.e. KEPT) has been developed. The main features of KEPT are:

- a unique and centralized Knowledge Repository (i.e. KR) collecting experience coming from the different CoPs involved in the whole production process with benefits from the knowledge maintenance and sharing point of view;
- a unique management tool (i.e. KRM) that allows distributed knowledge–based Knowledge Processing Modules (i.e. KPM) to work on dynamic views of KEPT KR;

Currently, KEPT contains a formal model of the knowledge concerning the Compounding and Tuning KBSs, the former developed as a rule–based system, the latter as a case–based tool. The other two modules of P–Truck, Mixing and Curing, work on a dedicated knowledge base that are going to be integrated into the general architecture of KEPT, in order to have a complete representation of all the knowledge involved in the tire production process.

The P–Truck system has been designed and developed according to a *Component Based* approach. KPMs have been designed in order to favor the integration of heterogeneous and interacting components, independent development of components, extendible architecture, separation between business and presentation logic, portability, integration on the existing information system and Web orientation. Future activities within the P–Truck project will concern the integration of already developed KPMs and KEPT according to the designed architecture and their deployment to Business Unit Truck of Pirelli.

## References

[1] Aamodt, A., Plaza, E., *Case–Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches*, AI Communications, Vol. 7, No. 1, pp. 39-59, 1994. 503, 507

[2] Akkermans, H., de Hoog, R., Shreiber, A., van de Velde, W.,Wielinga, B., *CommonKADS: A Comprehensive Methodology for KBS Development*, IEEE Expert, pp 28-37, 1994. 500

[3] Angele, J., Fensel, D., Studer, R., *Developing Knowledge-Based Systems with MIKE*, Journal of Automated Software Engineering, 1998. 500

[4] Bandini, S., Manzoni, S., *CBR Adaptation for Chemical Formulation*, in D. W. Aha, I. Watson (eds.), Case-Based Reasoning Research and Development, LNAI 2080, Springer-Verlag, Berlin, pp. 634-647, 2001. 506

[5] Bandini, S., Manzoni, S., Sartori, F., *Acquiring Knowledge and Numerical Data to Support CBR Retrieval*, in Gomez–Perez, A., Benjamins, R. (Eds.), Proceedings of the 13th European Knowledge Acquisiton Workshop (EKAW02), *Knowledge Engineering and Knowledge Management*, LNAI 2473, Springer Verlag, 2002. 500

[6] Bandini, S., Manzoni, S., Simone, C., *Tuning Production Processes through a Case Based Reasoning Approach*, in Craw, S., Preece, A., eds., LNAI/2416, Springer-Verlag, Berlin, 2002. 507

[7] Bergmann, R., Stahl, A., *Similarity Measures for Object- Oriented Case Representations*, Springer Verlag, LNAI 1488, 1998. 507

[8] Cair, O., *The KAMET Methodology: Contents, Usage and Knowledge Modeling*, in Gaines, B. and Mussen, M. (eds.), Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98), SRGD Publications, Department of Computer Science, University of Calgary, Proc-1, pp 1-20, 1998. 500

[9] Cussler, E. L., and Moggridge, G. D., *Chemical Product Design*, Cambridge University Press, ISBN 0521796334, Cambridge, 2001. 500

[10] Davenport, T., Prusack, L., *Working Knowledge - How Organizations Manage What They Know*, HBS Press, 1998. 499

[11] Genesereth,M. R., Ketchpel, S. P., *Software agents*, Communications of the ACM, Vol. 37 (7), pp 48-53, 1994. 502

[12] Hayes-Roth, F., Jacobstein, N., *The State of Knowledge Based Systems*, Communications of the ACM, 37(3) March 1994, pp 27-39. 500

[13] Jennings, N. R., *An agent-based approach for building complex software systems*, Communications of the ACM, Vol. 44 (4), pp 35-41, 2001. 502

[14] JavaTM 2 Platform, available at http://java.sun.com/j2ee/, 2002.

[15] Kolodner, J. *Case–Based Reasoning*, Morgan Kaufmann, San Mateo (CA), 1993. 507

[16] Manzoni, S., Mereghetti, P., *A Tree Structured Case Base for the System P–Truck Tuning*, UK CBR Workshop at Expert Systems 2002, Cambridge, 10 Dec. 2002, University of Paisley, Glasgow, pp. 17-26, 2002. 508

[17] Nonaka, I. and Takeuchi, H., *The knowledge-creating company: how Japanese companies create the dynamics of innovation*, Oxford University Press, New York, NY, 1995. 499

[18] Robertson, J., 2002, *Benefits of a KM Framework, Intranet Journal*, available at http://www.intranetjournal.com/articles/200207/*pse_07_31_02a.html* 500

[19] Santosus, M., Surmacz, J., *The ABCs of Knowledge Management*, available at http://www.cio.com/research/knowledge/edit/kmabcs.html, 2002. 499

[20] Wenger, E., *Community of Practice: Learning, meaning and identity*, Cambridge University Press, Cambridge, MA, 1998. 500, 507

# A CSP-Based Interactive Decision Aid for Space Mission Planning*

Amedeo Cesta, Gabriella Cortellessa, Angelo Oddi, and Nicola Policella

Planning & Scheduling Team
Institute for Cognitive Science and Technology
Viale K. Marx 15, I-00137 Rome, Italy
{cesta,corte,oddi,policella}@ip.rm.cnr.it
http://pst.ip.rm.cnr.it

**Abstract.** This paper describes an innovative application of AI technology in the area of space mission planning. A system called Mexar has been developed to synthesize spacecraft operational commands for the memory dumping problem of the ESA mission called Mars Express. The approach implemented in Mexar is centered on constraint satisfaction techniques enhanced with flexible user interaction modalities. This paper describes the effort in developing a complete application that models and solves a problem, and also offers functionalities to help users in interacting with different aspects of the problem. The paper surveys the design principles underlying the whole project and shows how different components contribute to the delivered system.

## 1 Introduction

Mars Express is a space probe launched by the European Space Agency (ESA) on June 2, 2003 that will be orbiting around Mars starting from the beginning of 2004 for two years. Like all space missions, this program generates challenging problems for the AI planning and scheduling community. Mission planning is a term that defines a complex set of activities aimed at deciding the "day by day" tasks on a spacecraft and at figuring out if spacecraft safety is maintained and mission goals are met on a continuous base. Supporting a complete mission planning problem is a quite challenging goal involving several sub-activities. One of such sub-activities, the dumping of the on-board memories to the ground station is the topic of a study which the authors have conducted for Mars Express.

A space system continuously produces a large amount of data which derives from the activities of its payloads (e.g. on-board scientific programs) and from on-board device monitoring and verification tasks (the so called *housekeeping data*). All these data, usually referred to as *telemetry*, are to be transferred to Earth during downlink connections. Mars Express is endowed with a single

* This work describes results obtained in the framework of a research study conducted for the European Space Agency (ESA-ESOC) under contract No.14709/00/D/IM.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 511–522, 2003.
© Springer-Verlag Berlin Heidelberg 2003

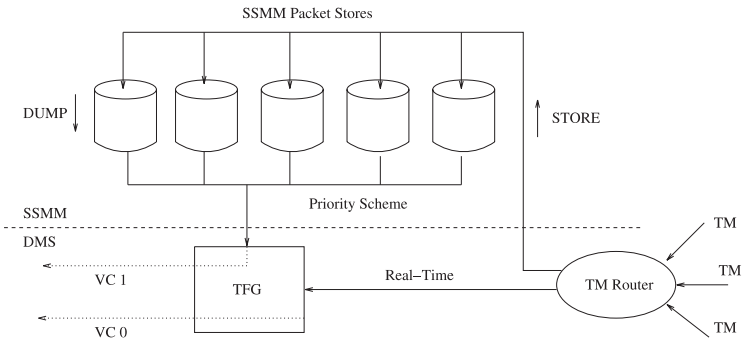



**Fig. 1.** On-board telemetry flow. The different telemetry (TM) data produced on board are stored on the on-board memory (SSMM) subdivided into packet stores. Memory stores are then downloaded in different dumps that transfer data to the ground

pointing system, thus during regular operations, it will either point to Mars and perform payload operations or point to Earth and transmit data through the downlink channel. As a consequence on-board data are first stored on the Solid State Mass Memory (SSMM) then transferred to Earth during *temporal visibility windows*. In Fig. 1 a sketchy view of the components on Mars Express that are relevant for this problem is presented. An effective management of on-board memory and a good policy for downlinking its data are very important for a successful operation of the spacecraft. The authors' study has addressed the problem of automatically generating downlink commands for on-board memory dumping. They have formalized the problem as the Mex-Mdp (the Mars Express Memory Dumping Problem), defined a set of algorithms solving this problem, and implemented an interactive system, called Mexar, which allows human planners to continuously model new Mex-Mdp instances, solve them, and inspect a number of the solution's features.

This paper gives an overview of this experience and in particular aims at demonstrating that for delivering complete applications containing AI technology an effort is needed to combine results from different AI lines of research. The approach described herein is centered on the formalization of the problem as a CSP (Constraint Satisfaction Problem) and on integrating a basic CSP representation with a module endowed with multi-strategy solvers and a second module devoted to interaction with human users. In Section 2 the Mex-Mdp is formalized, Section 3 introduces a general view of the CSP software architecture, Sections 4-6 describe the components of such an architecture as they are integrated in the Mexar system that has been delivered to ESA in May 2002. Some comments end the paper.

## 2 The Mars Express Memory Dumping Problem

The basic ontology to describe the Mex-Mdp domain focuses on two classes of objects: *resources* and *activities*. Resources represent subsystems able to give services, and activities model tasks to be executed on such resources. In addition, a set of *constraints* refines the relationships between the two types of objects. Three types of resources are modeled:

- *Packet Stores.* The on-board memory is subdivided into a set of separated packet stores $pk_i$ which cannot exchange data among each other. Each one has a fixed capacity $c_i$ and can be assigned a priority value to model different relevance of their data content. Each packet store, which can be seen as a fixed size file, is managed cyclically: when it is full the older data are overwritten. Within a packet store, data are segmented in data packets.
- *On-Board Payloads.* An on-board payload can be considered as a *finite state machine* in which each state has a different behavior in generating observation data (i.e., in each possible state the payload has a different generation data rate).
- *Communication Channels.* These resources are characterized by a set of separated communication windows identifying intervals of time for downlink. Each temporal window has a constant data rate.

Activities describe *how* resources are used. Each activity $a_i$ has an associated execution time interval, which is identified by its start-time $s(a_i)$ and end-time $e(a_i)$. Each activity is characterized by a particular set of resource requirements and constraints. Mex-Mdp includes three types of activities:

- *Payload Operations.* A payload operation $por_i$ corresponds to a scientific observation. Each $por_i$ generates a certain amount of data which is decomposed into different *store* operations according to the Mars Express operational modalities, and distributed over the set of available packet stores.
- *Continuous Data Streams.* The particular case of the continuous data stream operations $cds_i$ is such that $s(cds_i) = 0$ and $e(cds_i) = +\infty$ (where $+\infty$ is internally represented as a finite temporal horizon). This activity represents a continuous generation of data with a fixed average data rate (it is used to model housekeeping). Indeed we represent a *cds* as a periodic sequence of store operations. In particular, given *cds* with a flat rate $r$, we define a period $T_{cds}$, such that, for each instant of time $t_j = j \cdot T_{cds}$ $(j = 1, 2, \dots)$ an activity $st_{ij}$ stores an amount of data equal to $r \cdot T_{cds}$.
- *Memory Dumps.* A memory dump operation $md_i$ transfers a set of data from a packet store to a transfer device (Transfer Frame Generator, the TFG of Fig. 1). Those activities represent the transmission of the data through the communication channel.

Given a set of memory store operations from both the scientific observations $POR = \{por_1, por_2, \dots, por_n\}$ and the housekeeping $CDS = \{cds_1, cds_2, \dots, cds_m\}$ a *solution* is a set of dumping operations $S = \{md_1, md_2, \dots, md_s\}$ such that the following constraints are satisfied:

- The whole set of on board data are "available" on ground within a temporal horizon $\mathcal{H} = [0, H]$.
- Each dump operation starts after the generation of the corresponding data. For each packet store, the data are moved through the communication channel according to a FIFO policy.
- Each dump $md_i$ has an assigned time window $w_j = \langle r_j, s_j, e_j \rangle$, such that the dumping rate is $r_j$ and the constraint $s_j \leq s(md_i) \leq e(md_i) \leq e_j$ holds. Dump operations cannot reciprocally overlap.
- For each packet store $pk_i$ and for each instant $t$ within the considered temporal horizon, the amount of stored data has to be below or equal to its capacity $c_i$ (no overwriting is allowed).

The solutions should satisfy a quality measure. According to requirements from ESA personnel a high quality plan delivers all the stored data as soon as possible according to a definite policy or objective function. To build an effective objective function for this problem the key factor is:

- the turnover time of a payload operation $por_i$: $tt(por_i) = del(por_i) - e(por_i)$, where $del(por_i)$ is the delivery time of $por_i$ and $e(por_i)$ is the end time of the payload operation on board;

An objective function which considers this item is the *mean $\alpha$-weighted turnover time* $MTT_\alpha$ of a solution $S$:

$$MTT_\alpha(S) = \frac{1}{n} \sum_{i=1}^{n} \alpha_i \; tt(por_i) \tag{1}$$

Given an instance of a Mex-Mdp, an *optimal solution* with respect to a weight $\alpha$ is a solution $S$ which *minimizes* the objective function $MTT_\alpha(S)$. The weight $\alpha$ can be used to take into account two additional factors: the data priority and the generated data volume. In the experimental comparison of this paper, we use the *Mean Turnover Time* ($MTT$) with $\alpha_i = 1$, $i = 1..n$. For a more detailed description of the problem see [6] where it is also shown how the optimization problem for Mex-Mdp is *NP-hard*.

## 3 A Software Architecture for a CSP Representation

In our previous work we have developed a software framework for constraint-based scheduling called O-Oscar [3]. This framework has spawned a preliminary prototype of Mexar which has been used to deepen our comprehension of different aspects of the problem. A new CSP approach to the problem was then synthesized and an optimized version of Mexar was developed in Java and delivered to the ESA. It is worth noting that Mexar was developed using O-Oscar's same architectural approach, even if specific subcomponents were re-implemented. The general architecture we are referring to is composed of four modules which, as shown in Fig. 2, implement a complete CSP approach to problem solving:

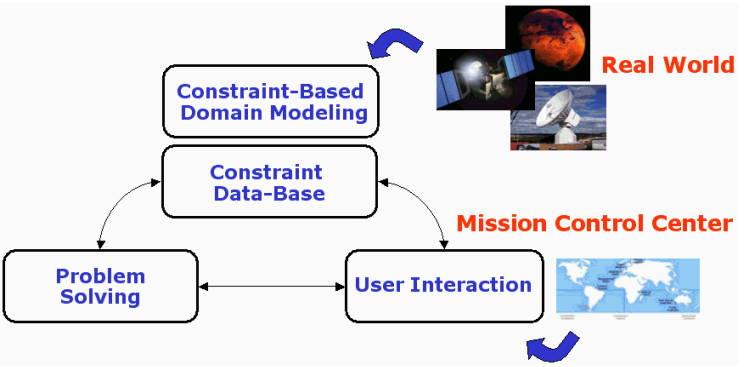



**Fig. 2.** Components of a CSP architecture

**Constraint-Based Domain Modeling.** This module models the real domain, captures the dynamic rules according to which the domain evolves and establishes a representation of the problem.

**Constraint Data Base.** The key part of a CSP architecture is the Constraint Data Base (CDB), a module all the others rely on. It provides data structures for representing the domain, the problem, the solution and its management as a set of constraints.

**Problem Solving.** This module is responsible for implementing the solution algorithms. In a planning and scheduling domain, we are not necessarily concerned with optimization problems, and approximations of the optimum solution are also acceptable. A solutions is often the result of an iterative process: an initial schedule is found, problems are identified, constraints are relaxed and changes are made on the solution. The Problem Solver captures this iterative process endowing the user with multiple algorithms he or she can choose among.

**User Interaction.** This module directly interacts with the user, and allows him or her to take part in the process of finding a solution by providing advanced problem solving functionalities. Along with the information which is fed back to the user, the interactive components deliver variable levels of control which range from strategy selection to iterative solution optimization.

This is a general and abstract picture of the approach to the problem. Indeed this structure underscores the different aspects involved in a complete approach and significantly impacted the subdivision of work during system development. It is worth noting that the first two modules are strictly interconnected, and will be described together in the following Section. They are separated in the figure to remark the existence of a basic "core" data structure, the constraint-data base, which is responsible for the integration of the other three functionalities.

## 4 A Constraint-Based Model for MEX-MDP

A CSP instance involves a set of variables $X = \{x_1, x_2, \ldots, x_n\}$ in which each element has its own domain $D_i$, and their possible combinations are defined by a set of constraints $C = \{C_1, C_2, \ldots, C_m\}$ s.t. $C_i \subseteq D_1 \times D_2 \times \cdots \times D_n$. A solution consists of assigning to each variable one of its possible values s.t. all the constraints are satisfied. A CSP representation of a problem should focus on its important features. In the case of MEX-MDP, we selected the following characteristics: (1) the temporal horizon $\mathcal{H} = [0, H]$, (2) the store operations that are characterized by their start time $t$ and their amount of data $d$, (3) the temporal windows in which no communication may occur, (4) the finite capacity $c$ of each memory bank, (5) the FIFO behavior of the memory banks.

Considering the first three items we split the temporal horizon $\mathcal{H}$ in different contiguous temporal windows according to *significant events*: store operations and change of transmission rate. The idea is to create a new window for each significant event on the timeline. This partitioning allows us to consider a temporal interval $w_j$ in which store operations do not happen (except for its upper bound) and the data rate is constant. Furthermore, the packet stores' behavior allows us to perform an important simplification. In fact, it is possible to consider both the data in input and those in output to/from the memory as flows of data, neglecting the information about which operations those data refer to (such information can be rebuilt with a straightforward post-processing step). Thus, the decision variables are defined according to the set of windows $w_j$ and to the different packet stores. In particular we consider as *decision variables* $\delta_{ij}$, the amount of data dumped from the packet store $pk_i$ within the window $w_j$. According to the partition in separate windows we introduce also: (a) $d_{ij}$, the amount of data stored in $pk_j$ at $t_i$, (b) $l_{ij}$, the available capacity of $pk_j$ at $t_i$, (c) $b_j$, the maximal dumping capacity within the window $w_j$. All these items represent the input of the problem.

A fundamental constraint captures the fact that for each window $w_j$ the difference between the amount of generated data and the amount of dumped data cannot exceed $l_{ij}$, the maximal imposed level in the window (*overwriting*). Additionally, the dumped data cannot exceed the generated data (*overdumping*). We define the following inequalities as *conservative constraints*.

$$\begin{aligned} &\sum_{k=0}^{j} d_{ik} - \sum_{k=1}^{j} \delta_{ik} \leq l_{ij} \\ &\sum_{k=0}^{j-1} d_{ik} - \sum_{k=1}^{j} \delta_{ik} \geq 0 \end{aligned} \qquad i = 1 \ldots n,\ j = 0 \ldots m \tag{2}$$

A second class of constraints considers the dumping capacity imposed by the communication channel. The following inequalities, called *downlink constraints*,

state that for each window $w_j$ it is not possible to dump more data than the available capacity $b_j$.

$$0 \leq \sum_{i=1}^{n} \delta_{ij} \leq b_j \quad j = 1 \ldots m \tag{3}$$

### 4.1 Domain Filtering Rules

The formalization of the problem as a CSP allows for the synthesis of *propagation rules* which examine the existing search state to find implied commitments. Though, in general, it is not possible to remove all inconsistent values through propagation rules, they considerably speed up the solving procedures. In Mex-Mdp two basic results allows the generation of propagation rules on the basis of the conservative and downlink constraints (proofs are omitted for lack of space). The first theorem proves that a minimal amount of data has to be dumped, considering the amount of data still stored in the packet store, otherwise the memory will not able to serve neither the present nor the future data. The second one states that the value for $\delta_{ij}$ cannot exceed the residual channel capacity.

**Theorem 1.** *For each decision variable $\delta_{ij}$, with $i = 1 \ldots n$, $p = 1 \ldots m$, and $j = 1 \ldots p$, the set of feasible values is contained in the interval:*

$$\delta_{ij} \in [\sum_{z=0}^{p} d_{iz} - l_{ip} - \sum_{w=1, w \neq j}^{p} ub_{\delta_{iw}}, \sum_{z=0}^{p} d_{iz} - \sum_{w=1, w \neq j}^{p} lb_{\delta_{iw}}]$$

**Theorem 2.** *The feasible values for each decision variable $\delta_{ij}$, with $i = 1 \ldots n$, and $j = 1 \ldots m$, are bounded by the following upper bounds:*

$$\delta_{ij} \leq b_j - \sum_{z=1, z \neq i}^{n} lb_{\delta_{zj}}$$

## 5 Problem Solving

A detailed description of the problem solving techniques is given in [6]; here we summarize the main results. To solve a Mex-Mdp represented as described in the previous section we have developed a two-stage approach:

**Data Dump Level.** We first figure out an assignment for the set of decision variables $\delta_{ij}$ such that the constraints from Mex-Mdp, see (2) and (3), are satisfied.

**Packetization Level.** The second stage is a constructive step: starting from the solution of the CSP derived at the first stage, the single data dumps within each of the windows $w_j$ are synthesized (that is, each $\delta_{ij}$ of the previous phase is translated into a set of dump activities).

MEXAR is endowed with a multi-strategy solver implemented on the basic blackboard represented by the Constraint Data Base. In particular, we have two algorithms, a greedy and a randomized algorithm, which compute new solutions for a MEX-MDP. A third, tabu search algorithm is used to look for local optimizations on a current solution obtained by the first two approaches. The *Greedy Algorithm* simply consists in assigning a value to each decision variable according to a heuristic. The variables are selected considering the windows in increasing temporal order. Two different solving priority rules are implemented: (a) CFF (Closest to Fill First) selects the packet store with the highest percentage of data volume. (b) HPF (Highest Priority First) selects the packet store with the highest priority. In case a subset of packet stores has the same priority, the packet store with the smallest store as outcome data is chosen. The *Randomized Algorithm* implements a basic random search [5] that turns out to be quite effective in this domain. This method iteratively performs a random sampling of the search space until some termination criteria is met. In our approach we select the variable in a random way, then the maximal possible value is assigned, considering: (1) the data contained in the packet store, (2) the amount of data already planned for dumping and (3) the dump capacity of the window. Both greedy and randomized algorithms are aided by the propagation rules. As usual in CSP, they allow to avoid inconsistent allocation and to speed up the search. The *Tabu Search* implements an instance of this well know local search procedure for this domain. A specialized move tries to improve the objective function $MTT_1(S)$ performing exchanges on data quantities between pairs of windows $w_j$. The current tabu is very basic, and further studies are under way to refine its effectiveness.

**Evaluation.** Table 1 contains a comparison of the application of the different algorithms to a set of MEX-MDP problems (for details on the benchmark generation and on lower bound synthesis see [4]). The values represent the percentage difference between the solution obtained by each approach and a lower bound value. It is possible to see that the greedy algorithm achieves low quality solutions and, additionally, cannot solve some instances. The randomized algorithm obtains higher quality solutions. The application of tabu search to both the greedy algorithm and the randomized one, produces, in general, an improvement of the solutions. Of course in the case of the randomized algorithm the improvement is smaller because the local search procedure already starts from high quality solutions which are more difficult to improve.

## 6 Designing a User Interaction for MEXAR

The idea we have pursued in the development of a support tool for the MEX-MDP is to endow a human planner with an advanced and helpful tool able to support and enhance his/her solving capabilities. Such a support tool should enable the user to more easily inspect a problem, analyze its features and find satisfactory solutions through step by step procedures, while guaranteeing continuous control

**Table 1.** Comparison among the solvers. Each column represents the percentage difference between the solution obtained with each approach w.r.t. a lower bound

| pr. # | pr. size | greedy | greedy+tabu | random sampling | rand+tabu |
|---|---|---|---|---|---|
| 1 | 96 | 265,32 | 124,19 | 30,87 | 29,30 |
| 2 | 91 | 281,98 | 123,58 | 33,21 | 30,41 |
| 3 | 86 | 303,21 | 131,90 | 33,46 | 31,53 |
| 4 | 76 | 326,71 | 133,36 | 33,77 | 32,64 |
| 5 | 87 | - | - | 18,79 | 18,40 |
| 6 | 81 | - | - | 20,73 | 19,09 |
| 7 | 71 | - | - | 23,00 | 22,50 |
| 8 | 58 | 69,90 | 69,90 | 17,35 | 16,96 |
| 9 | 32 | 2,90 | 2,90 | 0,92 | 0,92 |
| 10 | 28 | 3,15 | 3,15 | 1,20 | 1,20 |
| 11 | 243 | 14,42 | 14,05 | 5,05 | 4,88 |
| 12 | 230 | 14,72 | 14,34 | 5,15 | 4,69 |
| 13 | 226 | 13,00 | 12,61 | 4,72 | 4,54 |
| 14 | 217 | 12,98 | 12,59 | 4,76 | 4,20 |
| 15 | 204 | 13,26 | 12,86 | 4,83 | 4,42 |
| 16 | 15 | 23,94 | 23,94 | 10,14 | 9,73 |
| 17 | 12 | 30,73 | 30,73 | 12,64 | 12,17 |

over the problem solving process. It should indeed guarantee an easy interaction protocol, providing friendly representations of the problem, the solutions and all the entities of the domain, while hiding the underlying complexity so that the user can concentrate on higher level decision tasks. Figure 3 shows how these ideas have been actually implemented in MEXAR. A user is part of the real world and MEXAR endows him or her with an additional "lens" to analyze the world through the tool. To obtain this the CSP solving system —represented by the right box— is coupled with an Interaction Module —left box.

**Different Levels of Interaction.** The MEXAR Interaction Module has been designed subdividing the interaction capabilities in two layers. This is to provide different levels of participation in the solving process. A first layer allows the user to get acquainted with the problem and its features, optionally compiling

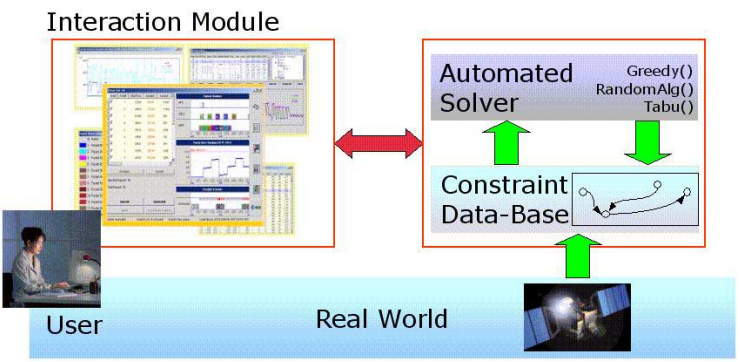
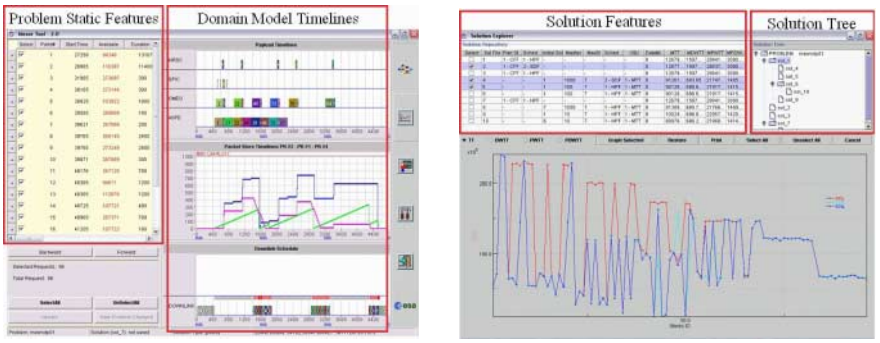



**Fig. 3.** The "human in the loop" schema




(a) Problem Analyzer (b) Solution Explorer

**Fig. 4.** User interaction in MEXAR

an initial solution. Once the human planner has a deeper knowledge of the problem, he/she can access the second interaction layer trying to contribute with his/her expertise and judgment to the problem solving. This mechanism makes it possible to choose between two modes of operation: to completely entrust the system with the task of finding a solution or to participate more interactively in the problem solving process. The different facilities are grouped together in two different graphic environments shown in Fig. 4, where additional squares and text have been inserted to underscore the meaning of subparts of the layouts:

**The Problem Analyzer.** The Problem Analyzer contains two groups of functionalities, one for problem editing and a second for problem solving. The interaction layout is based on the idea of "transparency" of all the different components of the representation of an MEX-MDP. This transparency follows what we call the "glass box principle": it enables the user to visually control the temporal behavior of the domain variables that are modeled. Figure 4(a) shows the basic idea used for visualizing a MEX-MDP problem and its solution. It basically represents different aspects of the problem (e.g. the POR list in textual form, a Gantt Chart representing their distribution over the payload timelines, the used capacity of the packet-stores). After having inspected the problem features a user can choose between different solving strategies to automatically solve the problem at hand. After the solver's work, a solution is displayed. Different perspectives of the solution are offered by providing alternative views of it: a table reconstructs all the details, while a graphic version of the solution, (Gantt Chart of the Dump Activities over the downlink channel), provides a more high level view of the solution.

**The Solution Explorer.** As we said before the Problem Solver allows a user to apply different solving methods to the same problem (greedy solver, randomized algorithm, local search, each with different possible tuning parameters).

In addition specific functionalities allow the user to save different solutions for the same problem and to guide a search for improvements of the current best result applying different optimization algorithms. The idea behind the Solution Explorer (see Fig. 4(b)) is that an expert user could try to participate more deeply in the problem solving process. A user might generate an initial solution, save it, try to improve it by local search, save the results, try to improve it by local search with different tuning parameters and so on. This procedure can be repeated for different starting points, resulting in the generation of different paths in the search space. Using the evaluation capability on a single solution and his or her own experience the user can visit the different saved solution series and the one which is most fit for execution. The current interface still does not completely abstract away the technicality of the algorithms from users. Further work is needed to fill the gap between the reasoning abilities of the user and the automated solver. Nevertheless, a step has been made to fill this gap by giving the user a supporting environment with different problem solving abilities while preserving the user's decisional authority.

**The System at Work.** Mexar has been delivered to ESA-ESOC in May 2002 and is currently available to mission planners. Users' reactions have been quite positive. We highlight in particular a real interest in the idea of using an automated tool that performs boring and repetitive task on their behalf which preserves the user's control on the flow of actions and the possibility to choose the final solution supported by the potentialities of the automated tool. In addition, the users' interest in the use of intelligent techniques for shortening the time needed to solve complex mission tasks is quite interesting. Mexar's functionalities for solution revision suggest a modality of work with a loop ⟨*user, automated support*⟩, which is currently under further investigation. In particular, the Solution Explorer provides a concept of human guided search (see also different approaches like [2]), which can be potentially very useful in all those applications where the combinatorics of the problems at hand are very high and the integration of competencies could be of great help in the search for a solution. Additionally, we are considering the addition of explanation functionalities to the system in order to increase users awareness of the problem solving cycle.

## 7 Conclusions

This paper has introduced Mexar, a complete software system grounded on AI techniques that solves a quite relevant sub-problem in space mission planning. The paper is focused on describing the general principles that are underlying the architectural development of Mexar, on technical work we have done on the CSP representation of the basic problem and on the interactive services that allow to develop an end-to-end solution for the human mission planners.

Before ending the paper, we would like to emphasize the key role of the integration of different AI techniques in the same system to obtain a complete

application. The flexibility of the CSP representation has turned out to be very useful. The declarative problem representation has allowed to realize different solvers but also to integrate effective interaction modalities. Additionally, it is worth noting the equal subdivision of our working effort between problem solving and user interaction. This second module, again, turned out to be of great importance in space applications dedicated to mission ground segments where the experience of the users should be taken constantly into account and integrated in the solving process. In this area the synthesis of mixed-initiative problem solvers is playing an increasingly important role in deployed applications (see [1] for another example).

## Acknowledgements

This work could not have been possible without the support at ESA-ESOC of both the project officer Fabienne Delhaise and the Mars Express mission planners Michel Denis, Pattam Jayaraman, Alan Moorhouse and Erhard Rabenau.

## References

[1] M. Ai-Chang, J. Bresina, L. Charest, A. Jonsson, J. Hsu, B. Kanefsky, P. Maldague, P. Morris, K. Rajan, and J. Yglesias. MAPGEN: Mixed Initiative Planning and Scheduling for the Mars 03 MER Mission. In *Proceedings Seventh Int. Symposium on Artificial Intelligence, Robotics and Automation in Space (i-Sairas-03), Nara, Japan, May*, 2001. 522

[2] D. Anderson, E. Anderson, N. B. Lesh, J. W. Marks, B. Mirtich, D. Ratajczack, and K. Ryall. Human-Guided Simple Search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 2000)*, 2000. 521

[3] A. Cesta, G. Cortellessa, A. Oddi, N. Policella, and A. Susi. A Constraint-Based Architecture for Flexible Support to Activity Scheduling. In *Proceedings of Italian Conference of Artificial Intelligence, AI*IA 01*, 2001. 514

[4] A. Cesta, A. Oddi, G. Cortellessa, and N. Policella. Automating the Generation of Spacecraft Downlink Operations in Mars Express: Analysis, Algorithms and an Interactive Solution Aid. Technical Report MEXAR-TR-02-10 (Project Final Report), ISTC-CNR [PST], Italian National Research Council, July 2002. 518

[5] S. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. 518

[6] A. Oddi, N. Policella, A. Cesta, and G. Cortellessa. Generating High Quality Schedules for Spacecraft Memory Downlink Problem. In *Principles and Practice of Constraint Programming,* $9^{th}$ *International Conference, CP 2003*, Lecture Notes in Computer Science. Springer, 2003. 514, 517

# E-mail Categorization, Filtering, and Alerting on Mobile Devices: The ifMail Prototype and its Experimental Evaluation

Marco Cignini, Stefano Mizzaro, and Carlo Tasso

Department of Mathematics and Computer Science, University of Udine
Via delle Scienze, 206, Udine, Italy
{cignini,mizzaro,tasso}@dimi.uniud.it
http://www.dimi.uniud.it/~{cignini,mizzaro,tasso}

**Abstract.** We propose an integrated approach to email categorization, filtering, and alerting. After a general introduction to the problem, we present the ifMail prototype, capable of: categorize incoming email messages into pre-defined categories; filter and rank the categorized messages according to their importance; and alert the user on mobile devices when important messages are waiting to be read. The second part of the paper describes an extended evaluation of the ifMail prototype, whose results show the high effectiveness levels reached by the system.

## 1 Introduction

*Email overload* is an important facet of information overload. Electronic mail, historically one of the first services made available by the Internet to the large public audience, is today one of the major activities of Internet users. All of us rely on email as one of the primary communication method, both at work and at home: email has, at least partially, supplanted paper mail, messages, and telephone conversations. The main problem is that the average user receives dozens of messages per day, and the trend is not slowing down at all [23].

Moreover, email software tools (Eudora, Outlook, Mozilla, to name just a few) are used not only in the standard ways foreseen by email tools designers, i.e., for reading and answering messages, but also in more "perverted" ways. We refer here to archiving, managing a personal agenda or serving as a reminder tool: people send mail to themselves as a reminder; people use the inbox message list as an agenda; people use email for task management and delegation; people hit reply for avoiding to type in a long list of addresses; people archive a whole message when the attachment is an important document; people use email as a file transfer mean; and so on. Because of this creative use of email, another meaning for the "email overload" expression (the overloading of uses of this tool) has arosen [23], and email has been named a serial-killer application [8].

Also, usage of email is a highly personalized activity, and people use email in amazingly different ways. People read emails with different strategies: *archivers*

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 523–535, 2003.
© Springer-Verlag Berlin Heidelberg 2003

try to read everything and not miss anything important, and *prioritizers* want to limit the time spent on email reading to switch to "real work" [9]. Accordingly to Whittaker and Sidner [24], people can be divided into *no filers* (that keep all the messages in their inbox), *frequent filers* (that constantly clean up their inbox), and *spring cleaners* (that clean up their inbox once every few months). Some of us are lucky and receive a manageable number of email messages per day, whereas others are completely overwhelmed. Unsolicited email, usually called spam or junk-mail, is constantly and worryingly increasing.

Finally, alerting is rather neglected: having only a visual and/or acoustical "You have new mail" notification on our desktop is a rather poor way of communication. Notifications could and should be delivered on the nowadays widely available smaller and portable devices (cell phones, PDAs, pagers, and so on), that are enabled to various network connection modes (GSM, GPRS, UMTS, Wi-Fi, Bluetooth, etc.), with the most appropriate modality (WAP-push, SMS, etc.). Notification should be done depending on features of the received messages like their number, their importance, the category they pertain to, and so on.

In this scenario, end users desperately need advanced tools for email processing, i.e., threading, categorizing, archiving, filtering, alerting, and so on. Today's email clients provide these functions in a rather limited way. All email tools notify the user sitting in front of his/her desktop that new mail has arrived by visual and/or acoustic messages. Mail tools allow manual categorization of messages (usually by drag-and-drop in one of a hierarchy of folders). A priority flag can be manually attached to a message by the sender, and shown to the receiver by the mail client. Filters based on pattern matching rules on (mainly) the structured part of messages (i.e., subject, sender, date, priority, size, etc.) can be manually defined by the user to filter out spam, to automatically move the received messages in the appropriate folder, and so on. All these activities are both time consuming and rather ineffective: manually defining a filter and managing a set of several filters puts a higher cognitive load to a user engaged in other activities and, often, the decision whether a message is interesting, junk, belonging to a certain category, and so on cannot be taken only on the basis of the structured part of the message but it has to be taken also on the basis of message body, attachment, meaning, and even context (i.e., the thread to which the message belongs, the current situation in which the user is, and so on).

The coming of portable devices is a new and important variable to add. Alerting will have an increased importance, and new mail tools and protocols might be designed to allow the user (both as a sender and as a receiver) to specify (manually, semi-automatically, or automatically) the alerting modalities of certain message categories. The well known limitations on bandwidth, screen size, and user cognitive load (time, distraction level, and so on) make extremely important to have a selective alerting functionality, capable of notifying the user only when really important messages arrive. Complex engineering solutions are needed because the limited computational power available on the mobile devices require a server side based solution, in which most of the computation takes place on the server. And, finally, the increased email access by mobile devices

will change the way people use email: nobody can predict all the range of new "perverted" or "creative" usages that mobile device users could imagine and adopt when mobile email tools will be broadly available (e.g., the sending of email to oneself as a remainder is likely to become much more frequent).

One of the research challenges today is to improve and make (at least partially) automatic the tasks of email categorization, filtering, and alerting. In this paper we deal with these research issues. The paper is structured as follows. In Section 2 we highlight the main issues related to email categorization and filtering and briefly survey the literature. In Section 3 we describe the ifMail prototype, from both conceptual and technical perspectives. In Section 4 an extended experimental evaluation of the effectiveness of our approach is presented. Section 5 closes the papers and sketches future developments.

## 2 Categorization and Filtering of Email Messages

Text categorization (or classification) is the grouping of documents into predefined categories [19]. State-of-the-art classifiers automatically built by means of machine learning techniques show an effectiveness comparable to manually built classifiers.

Email messages are very heterogeneous. Examples of variables that can range over rather wide set of values are: length, language(s) used, importance of the contained information, presence/absence of attachments of various kinds, formal/informal tone, emoticons, jargon. Also structured data contained in the header like date, sender, subject, number of recipients, are bound to wide variations. Given the peculiar nature of email messages, email categorization is a very particular case of general text categorization.

Various approaches, mainly derived from the experiments on generic text categorization, have been applied to email categorization [7]: Cohen [6] uses the RIPPER algorithm; Payne and Edwards [16] compare CN2 (a rule induction algorithm) with IBPL1 (a modified version of K-nearest Neighbor algorithm using memory based reasoning); Rennie [17] exploits naïve Bayes classifiers; Segal and Kephart [20] develop a system for semi-automatic categorization (i.e., the system proposes to the user three alternative folders for each message) based on TF-IFD; Brutlag and Meek [4] compare Linear Support Vector Machine, TF-IDF, and Unigram Language Model, and obtain that no method outperforms the others. All these approaches show rather similar results, with accuracy (percentage of messages classified in a correct way) around 70%-80%. An even more difficult problem, the clustering of email messages (i.e., given a set of email messages, extract the categories and classify the messages in the found categories), is tackled in [10].

Spam (or junk) email filtering has seen an increasing interest in last years, due to the increasing amount of unsolicited emails: Pantel and Lin [15] and Sahami et al. [18] exploit naïve Bayes classifiers; Adroutsopoulos et al. [1] use a memory-based (or instance-based) approach, implemented as a variant of the

K-nearest neighbor (K-nn) algorithm; Carreras et al. [5] rely on the boosting algorithm AdaBoost to find a good classification rule by combining weak rules.

Anti-spam filtering has been approached as a separate problem from email categorization, even if, at first glance, it seems just a 2-categories categorization problem. However, anti-spam is an easier problem than categorization not only because it handles just two categories, but also because the two categories are rather well defined (it is rather easy to define spam), clear-cut (it is rather easy to sort out spam from non-spam), and objective (usually, what is spam for one user is spam for everybody). In turn, email categorization is highly subjective: each user can choose rather different criteria for creating the categories (e.g., messages can be divided on the basis of the sender, of the topic, of one's own a-priori categorization of his job activity, and so on); the number of categories can vary a lot among users; the categories are sometimes not well defined (users can be very well organized or completely chaotic); and so on. Therefore, it is quite likely that a single fit-for-all email categorizer is not feasible, and that hybrid approaches are needed. Indeed, even if it is difficult to have a definitive comparison between the effectiveness of anti-spam filters and of email categorizers because of the high differences in the collections used, in the number and features of categories, and so on, it is evident that anti-spam filters effectiveness is rather higher (95% precision) than the more general email categorization problem.

The alerting problem is much less studied than email categorization and filtering: further research in terms of notification modalities, prototype implementation and evaluation, and user studies is needed.

The evaluation of the effectiveness of an email tool is not simple at all. The most naïve approaches show several limitations. Relying on general test collection like TREC (http://trec.nist.gov/) is not adequate, since the peculiar nature of email makes an email message different from a generic document. Usenet news seem more similar, but again differences do exist (e.g., an email message body usually starts with the name of the recipient, whereas this is obviously less frequent for Usenet messages). Privacy is also an important issue: since email messages contain private data, few people are willing to make public their messages; perhaps those people will anyway erase some of the more confidential messages, thus making available only a portion of their message archive, that would be a biased sample; anyway, people willing to make public their email archives are not a good sample for sure, since more reserved people are left out; and relying on message archives of mail lists leads again to a biased sample.

## 3 The ifMail Prototype

At the Udine University we have started to study some of the above described issues and, on the basis of our work in the last 10 years, we have developed the ifMail prototype. ifMail handles, with a content based approach, categorization, filtering of email messages, and alerting on mobile devices. ifMail overall operation is as follows. The messages in the incoming stream are processed to extract the internal representations used in subsequent steps. The internal representation contains term/weight (weight representing the importance of each term)

pairs, corresponding to both the structured part and the body of the email message. Categorization is obtained on the basis of a profile attached to each user-defined folder and dynamically updated by means of user's feedback. The profile contains two parts: a frame for the information included in the structured part of email messages, and a semantic network for the conceptual content of the body of messages [12]. The profile is matched with the internal representation of the incoming messages and the message is classified accordingly to its content. The matching takes into account both the structured and unstructured parts of email messages. Filtering, performed by re-using the evaluation made in the categorization phase, singles out the most relevant messages in each folder and alerting takes charge of notifying these messages to the user's mobile device. Our notion of filtering is therefore more general than just anti-spam filtering: ifMail tries to associate to each message a numeric figure representing the importance that the message has for the user.

ifMail categorization and filtering are based on the IFT (Information Filtering Tool) system [11,12], capable of profile building, storing, and matching. IFT has been developed on the basis of the UMT (User Modeling Tool) shell [3] and has been applied to a variety of systems and domains, e.g., Web filtering [2], filtering of enterprise documents [21], and filtering of scholarly publications [13]. IFT matches the profile associated to each category with the internal representation of each message and returns a result made up of three values: (i) *Coverage*, i.e., the percentage of the most relevant concepts of the profile which are also present in the documents, computed considering also the weights; (ii) *Match*, i.e., a measure of how much the concepts of the profile are present in the document (if they are more or less numerous in the document); and (iii) *Rank*, i.e., a synthetic value (ranging from 0 to 5), which is obtained as a combination of the previous two values. Categorization is performed on the basis of all three values; filtering is based on Rank score only.

Fig. 1 shows the overall architecture of ifMail. The main modules are:

- WebMail, that allows the user to access email functionalities via a Web browser. It has been developed specifically for this project in order to connect and integrate categorizing, filtering, and alerting. More specifically, the WebMail module implements the only user interface of the system and it allows the configuration of the innovative services.
- Mail Filtering and Classification Engine, made up by three sub-modules:
  a) Monitoring Agent, that monitors the arrival of new messages and calls the categorization and filtering operations. ifMail supports POP and IMAP servers, and any number of email accounts.
  b) Internal Representation Builder, that parses the text of message subject and body, extracts lexical tokens, removes stop words, extracts the stem of the terms, and builds the internal representation of the message, stored in the Internal Representation Database.
  c) Categorization, that executes categorization and handles feedback data. This module contains the IFT submodule: IFT compares the internal representation of the incoming message with each category profile, and modifies the category profile according to user's relevance feedback.

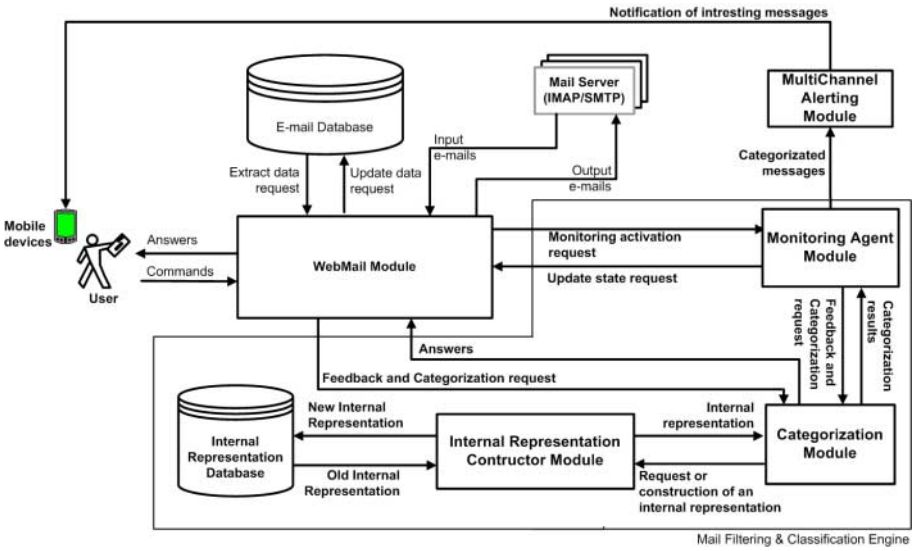



**Fig. 1.** ifMail overall architecture

– Multi Channel Alerting, that, on the basis of the categorization results and of user's personalized settings, notifies immediately to the user the most relevant messages via a mobile device.

Fig. 2 shows a snapshot of ifMail Web user interface: a quite standard email interface that allows standard mail management and that provides the commands and visualization items relevant to the new categorization and filtering features. The number of stars associated to each message is given by the Rank score associated to the message. The PDA screenshots in Fig. 2 show the multi-channel alerting of ifMail: in the screenshot on the left, the notification of the arrival of a new relevant message for the "myWork" category is shown. The user can detect (by the number of stars) the message relevance computed by the system, he can archive the message, read message data like sender and subject, or read the whole message body (screenshot on the right).

## 4 Experimental Evaluation

We have discussed in Section 2 the intrinsic limitations in the evaluation of advanced email tools, and some of the issues that make the evaluation of these tools a difficult task. In order to overcome these limitations, we have designed and carried out an extensive evaluation of the ifMail prototype, taking also into account previous experimental work carried out in recent years in our laboratory. The goal of the experimental activity has been the evaluation of categorization, filtering, and alerting capabilities of ifMail. We have run various simulations on 6 collections of email and newsgroups messages. We use the term "simulation"

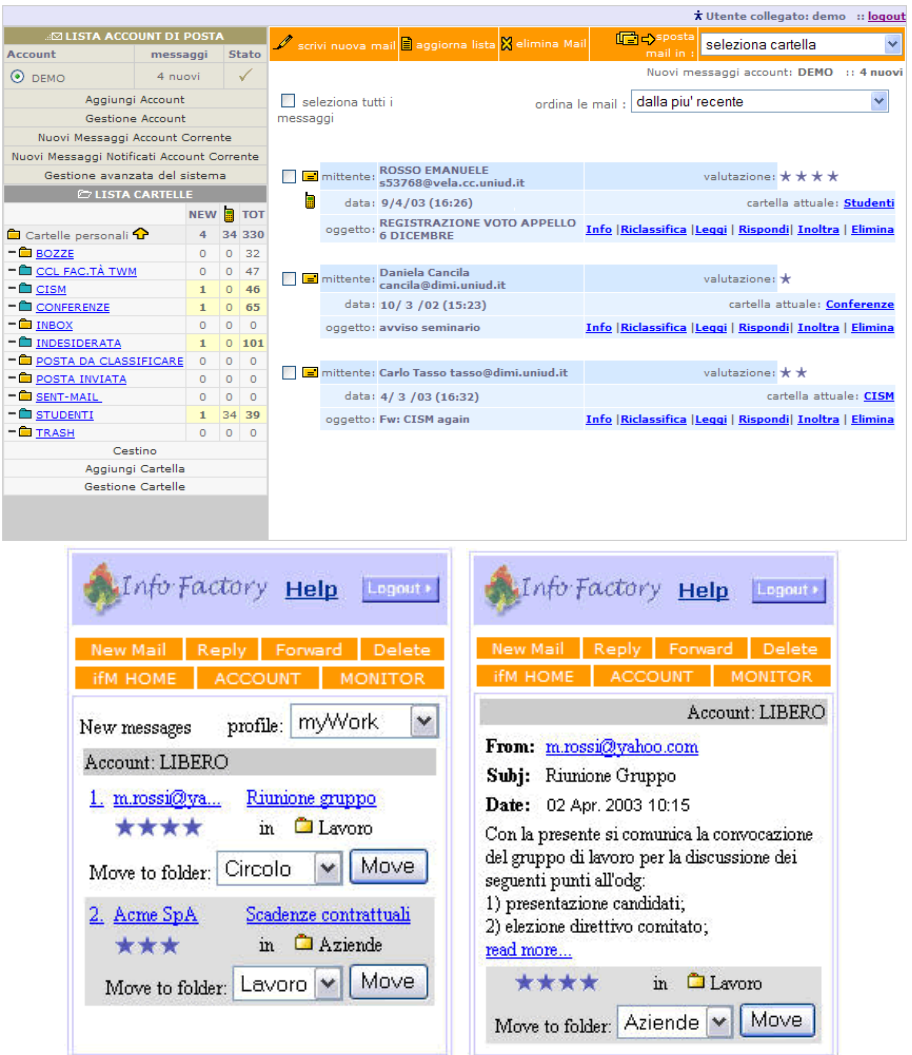



**Fig. 2.** ifMail user interface for Web mail (above) and PDA (below)

since the experiments have been performed in a simulated environment in which the typical actions that a user could perform on ifMail can be repeated at will, without engaging (and overloading) real users. Obviously, with this approach, we have intentionally not evaluated the usability of the user interface, nor we wanted to claim the effectiveness of our system in absolute terms. On the other hand, given the early development stage of the ifMail prototype, we were interested in evaluating some design decisions and in harvesting an experimental set of real

**Table 1.** Email message collections used in the experiments

| Message kind | Collection | Number of categories | Total number of messages |
|---|---|---|---|
| **Personal messages** | **A** | 9 | 540 |
| | **B** | 7 | 645 |
| **Newsgroups messages** | **C** | 7 | 525 |
| | **D** | 6 | 450 |
| | **E** | 7 | 540 |
| | **F** | 16 | 1309 |

data with a quick, light, and formative evaluation, capable of giving us hints on how to proceed with the development of the system.

Tab. 1 provides basic data on the six collections of email messages we have exploited: two of them come from real users, and include *all* the messages received over a period of about 30-40 days. Both users defined a set of categories (folders), to be used for evaluating the classification capabilities. The collections extracted from newsgroups concern a similar number of messages and categories, with the exception of collection F, which is significantly larger and was considered for evaluating whether the results obtained with similar collections (A through E) were maintained in a much heavier situation.

We have defined two different modes of operation of ifMail usage:

- Mode *One-by-one*, in which ifMail provides only an advice: the user reading a message is shown a hint on which category(ies) are likely to be its correct destination. By confirming or not confirming on each single message the (automatically) proposed categorization, the user provides relevance feedback, exploited by the system to update the relevant category profiles.
- Mode *Session*, in which ifMail automatically categorizes all the messages received during the current day (we have assumed daily batches of fixed size including 15 messages per day). The user provides relevance feedback only after all these categorizations have been done.

A first set of experiments concerned the comparison of these two modes of operation. The profiles associated to each folder were initially empty, and were incrementally built only through relevance feedback. Tab. 2 illustrates the average (over all the available collections) of precision, recall, and F1 measure [22, 25], where the results obtained for each category are combined using the micro-average indicator [19].

First of all, we notice that the values obtained are in the range from 70% to 80%. Other experiments reported in the literature [14, 19] concern the categorization of the Reuters-22713 collection (21.450 articles divided into 135 categories) or the Reuters-21578 collection (12.902 articles divided into 90 categories): the values obtained for the F1 measure are in the same 70%–80% range. We have considered this result as a confirmation of the adequacy of the baseline performance of ifMail. Furthermore, the values reported in Tab. 2 are average values,

**Table 2.** Comparison between session mode and one-by-one mode

| | Session Mode | One-by-one Mode |
|---|---|---|
| **Average Precision** | 75% | 79% |
| **Average Recall** | 72% | 76% |
| **Average F1** | 74% | 74% |

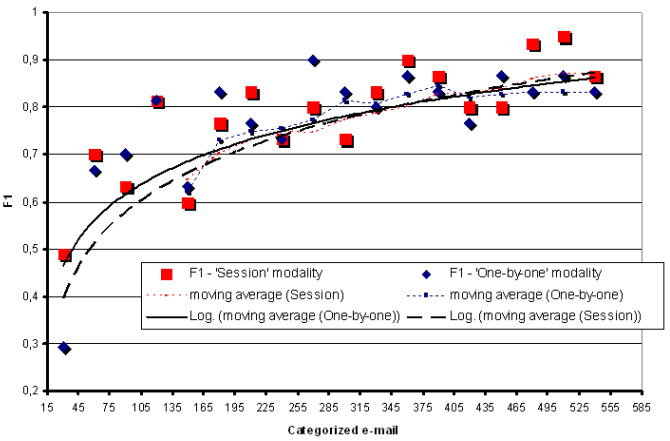



**Fig. 3.** Microaverage F1 in both operation modes for collection E

which include also the initial phases, where errors are most likely to happen: saturation ("steady state") values can be significantly higher.

Secondly, precision reaches higher levels than recall. We can interpret this phenomenon as follows: the number of messages considered is capable of reducing the number of categorization errors, but, on the other hand, is not sufficient for building profiles that cover all the concepts included in a category (and some message are not categorized, i.e. not assigned to any category). Finally, one-by-one mode outperforms session mode, reaching almost 80% in all the three considered indicators.

With reference to the same experiment, Fig. 3 shows the evolution (over the sequence of daily sessions and only for collection E) of the F1 measure. Both modes of operation reach values above 80%. The 70% level (conventionally indicating the termination of the initial learning phase) is reached earlier in the one-by-one mode. In the long run the two modes of operation reach the same level of performance.

Collections A and B, provided by real users, contained a Spam category, defined by the two users in order to collect all the "not desired" messages (typically unsolicited advertising). Considering the Spam folder of collection B, precision reaches more than 95% and recall the range 70%-80%: this can be explained by

**Table 3.** Results for categories with well defined topics

| Collection | Folder | Precision | Recall | F1 |
|---|---|---|---|---|
| **A** | *News* | 0.91 | 0.83 | 0.87 |
| **B** | *Students and courses* | 0.94 | 0.93 | 0.93 |
| | *Department news* | 0.85 | 0.91 | 0.88 |
| | *Seminars* | 0.86 | 0.91 | 0.88 |
| **C** | *ADSL* | 0.92 | 0.92 | 0.92 |

the fact that when a spam message is received, all the subsequent messages concerning the same topic will be detected, while new spam topics are not known since never seen before, so they are left in the inbox, i.e., not categorized. This highlights a significant advantage of our content-based approach to spam detection, in comparisons with standard anti-spam systems based on an archive of spam messages: our system can detect any new spam message which concerns topics that previously have been already classified as spam, independently from other facts (sender or subject already encountered or not).

Another (expected) phenomenon observed in the experimentation concerns the relationship between performance and specificity level of a category: whenever a category includes a well defined and limited topic, performance in terms of precision and recall is higher, reaching for both indicators the level of 85%. Analogously, for such categories, the learning phase is shorter. Tab. 3 illustrates such a situation for some categories with this characteristics.

Other experiments have been focused on identifying the best threshold to be employed for alerting. We have seen that using only the Rank value (an integer ranging from 1 to 5), precision was maximized (over 80%) and that, by increasing the specific value considered for the threshold, precision was further improved.

Finally, we have computed a measure of the effort required to the user of ifMail, in terms of the number of "move operations" of a message towards its correct folder (category). More specifically, we have considered successive groups of 60 messages (i.e., four days), and we have counted (Fig. 4) the number of correct system categorization operations and the number of user moves, i.e., the explicit indication done by the user on a single message, since the system was not able to categorize the message correctly. It is interesting to see that, as the user "teaches" to the system how to categorize, the system "learns". After about 70 messages received, the user needs to move about 50% of the messages to their correct folder. After about 300 messages, the system "has learned", and it is able to categorize correctly more than 50 messages out of the incoming 60, with a missed-categorization rate of less than 16%.

## 5 Conclusions and Future Work

We have discussed email categorization, filtering, and alerting. After a general introduction to the problem and a brief literature survey, we have presented the ifMail prototype, capable of: categorize incoming email messages into pre-

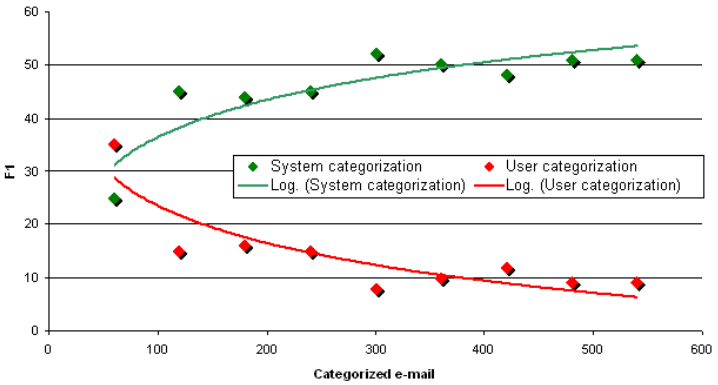



**Fig. 4.** Comparison of the number of user and system categorization actions

defined categories; filter and rank the categorized messages according to their importance; and alert the user on mobile devices when important messages are waiting to be read. We have also performed an extended evaluation of the ifMail prototype. The results show the high effectiveness levels reached by the system.

We will continue this research in various ways. We are currently improving the ifMail prototype and we plan a more complete evaluation after these improvements. We intend to deal with privacy issues with a novel approach, by implementing a software capable of analyzing the email archives of users by running on their computers and simulating the behavior of a categorization algorithm. The algorithm results should then be compared with the hand-made categorization and only the comparison results are made public. This software should be open source (to guarantee the privacy) and could be designed as a framework capable of hosting any categorization algorithm conforming to some well defined specifications. To take into account the time characteristics of messages (how long a message has been staying in the inbox, how long it has been in the unread status, how much time the user spent in reading it, or in answering it, for how long the user has not been checking his/her email, and so on) the software should also be capable of monitoring user's activity for a period of time.

## References

[1] I. Androutsopoulos, J. Koutsias, K. V. Chandrinos, G. Paliouras, and C. D. Spyropoulos, An Evaluation of Naive Bayesian Anti-Spam Filtering. *Proc. of the Workshop on Machine Learning in the New Information Age*, 11th European Conference on Machine Learning (ECML), pp. 9-17, Barcelona, Spain, 2000.

[2] F. A. Asnicar., M. Di Fant, C. Tasso User Model-Based Information Filtering. In *AI*IA 97: Advances in Artificial Intelligence - Proc. of the 5th Congress of the AI*IA*, Springer Verlag, Berlin, LNAI 1321, 1997, pp. 242-253, 1997.

[3] G. Brajnik, C. Tasso, A shell for Developing Non-Monotonic User Modeling System, *International Journal of Human-Computer Studies*, vol. 40, pp. 31-62, 1994.
[4] C. Brutlag, J. Meek, Challenges of the email domain for text classification, In *Proc. of the 17th International Conference on Machine Learning*, pp. 103-110, 2000.
[5] X. Carreras, L. Marquez, Boosting Trees for Anti-Spam Email Filtering, *Proc. of RANLP-01, 4th International Conference on Recent Advances in Natural Language Processing*, Tzigov hark, BG, 2001.
[6] W. Cohen, Learning Rules that Classify E-Mail, *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pp. 18-25, 1996.
[7] E. Crawford, J. Kay, and E. McCreath, Automatic Induction of Rules for e-mail classification, *Proc. of the Sixth Australian Document Computing Symposium*, Coffs Harbour, Australia, 2001.
[8] N. Ducheneaut and V. Bellotti, Email as Habitat. *Interactions*, Sept./Oct. 2001.
[9] W. Mackay, Diversity in the Use of Electronic Mail: A Preliminary Inquiry. *ACM Transactions on Office Information Systems*, 6(4), 380-397, 1988
[10] G. Manco, E. Masciari, M. Ruffolo, and A. Tagarelli, Towards An Adaptive Mail Classifier, *Atti dell'Ottavo Convegno AI*IA 2002*, Siena, Italy, p. 63, 2002.
[11] M. Minio, C. Tasso, User Modelling for Information Filtering on Internet Services: Exploiting an Extended Version of the UMT Shell, *UM96 Workshop on User Modeling for Information Filtering on the WWW*, Kaiula-Kona, Hawaii, USA, 1996.
[12] M. Minio, C. Tasso, IFT: un'Interfaccia Intelligente per il Filtraggio di Informazioni Basato su Modellizzazione d'Utente, *AI*IA Notizie IX(3)*, 21-25, 1996.
[13] S. Mizzaro and C. Tasso, Ephemeral and Persistent Personalization in Adaptive Information Access to Scholarly Publications on the Web. In *Adaptive Hypermedia and Adaptive Web-Based Systems, 2nd International Conference, LNCS 2347*, Malaga, pp. 306-316, 2002.
[14] I. Moulinier, G. Raskinis, J. G. Ganascia, Text Categorization: a Symbolic Approach, In *Proc. of the 5th Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, pp. 87-99, 1996.
[15] P. Pantel, D. Lin, Spamcop: A spam classification & organization program, in *Proc. of AAAI-98 Workshop on Learning for Text Categorization*, pp. 95-98, 1998.
[16] T. Payne, P. Edwards, Interface agents that learn: An investigation of learning issues in a mail agent interface, *Applied Artificial Intelligence*, Vol. 11, pp. 1-32, 1997.
[17] J. D. M. Rennie, ifile: An application of Machine Learning to E-Mail Filtering, In *Proc. KDD00 Workshop on Text Mining*, Boston, 2000.
[18] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz, A bayesian approach to filtering junk e-mail, in *AAAI-98 Workshop on Learning for Text Categorization*, 1998.
[19] F. Sebastiani, Machine Learning in Automated Text Categorization, *ACM Computing Surveys*, 34(1), 1-47, 2002.
[20] R. B. Segal, J. O. Kephart, Incremental Learning in SwiftFile, *Proc. of the International Conference on Machine Learning*, San Francisco, pp. 863-870, 2000.
[21] C. Tasso and M. Armellini, Exploiting User Modeling Techniques in Integrated Information Services: The TECHFINDER System. *Proc. of the 6th AI*IA Congress*, Bologna, I, September 14-17, 1999, Pitagora Editrice, pp. 519-522, 2000.
[22] K. Van Rijsbergen, *Information Retrieval*, 2nd ed. Butterworths, London, UK, 1979. http://www.dcs.gla.ac.uk/Keith/pdf.

[23] G. Venolia, L. Dabbish, J. J. Cadiz, and A. Gupta, Supporting Email Workflow. Microsoft Research Tech Report MSR-TR-2001-88, 2001.
[24] S. Whittaker and C. Sidner, Email Overload: Exploring Personal Information Management of Email. *Proc. of the ACM CHI Conference*, pp. 276-283, 1996.
[25] Y. Yang, X. Liu, A re-examination of text categorization methods, *Proc. of the 22nd ACM SIGIR Conference*, Berkley, CA, August 15-19, pp. 42-49, 1999.

# Applying Artificial Intelligence to Clinical Guidelines: The GLARE Approach

Paolo Terenziani[1], Stefania Montani[1], Alessio Bottrighi[1], Mauro Torchio[2], Gianpaolo Molino[2], Luca Anselma[3], and Gianluca Correndo[3]

[1] DI, Univ. Piemonte Orientale "A. Avogadro"
Spalto Marengo 33, Alessandria, Italy
[2] Lab. Informatica Clinica, Az. Ospedaliera S. G. Battista
C.so Bramante 88, Torino, Italy
[3] Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, Torino, Italy

**Abstract.** In this paper, we present GLARE, a domain-independent system for acquiring, representing and executing clinical guidelines. GLARE is characterized by the adoption of Artificial Intelligence (AI) techniques at different levels in the definition and implementation of the system.
First of all, a high-level and user-friendly knowledge representation language has been designed, providing a set of representation primitives.
Second, a user-friendly acquisition tool has been designed and implemented, on the basis of the knowledge representation formalism. The acquisition tool provides various forms of help for the expert physicians, including different levels of syntactic and semantic tests in order to check the "well-formedness" of the guidelines being acquired.
Third, a tool for executing guidelines on a specific patient has been made available. The execution module provides a hypothetical reasoning facility, to support physicians in the comparison of alternative diagnostic and/or therapeutic strategies. Moreover, advanced and extended AI techniques for temporal reasoning and temporal consistency checking are used both in the acquisition and in the execution phase.
The GLARE approach has been successfully tested on clinical guidelines in different domains, including bladder cancer, reflux esophagitis, and heart failure.

## 1 Introduction

Clinical guidelines represent the current understanding of the best clinical practice, and are now one of the most central areas of research in Artificial Intelligence (AI) in medicine and in medical decision making (see, e.g. [5, 7, 8, 12]). Clinical guidelines play different roles in the clinical process: for example, they can be used to support physicians in the treatment of diseases, or for critiquing, for evaluation, and for education purposes. Many different systems and projects have been developed in recent years in order to realize computer-assisted management of clinical guidelines (see e.g., Asbru [15], EON [10], GEM [16],

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, pp. 536–547, 2003.
© Springer-Verlag Berlin Heidelberg 2003

GLARE [19, 20, 21], GLIF [11], GUIDE [14], ONCOCIN [22], PROforma [4], T-HELPER [9], and also [6, 2]).

The overall challenge of designing and implementing such tools is very complex. In this paper we show how in the GLARE system the adoption of AI techniques provides relevant advantages, especially from the point of view of the user-friendliness of the approach (a more detailed description of GLARE's basic features can be found in [20]). GLARE's architecture is sketched in section 2. In section 3, we highlight GLARE's representation formalism. Section 4 and section 5 describe the acquisition tool and the execution tool functionalities respectively, with specific attention to the treatment of temporal constraints. Section 6 sketches some testing results. Finally, section 7 presents comparisons and conclusions.

## 2 Architecture of GLARE

The overall GLARE's architecture is a three-layered one (see figure 1).

The highest layer (*system layer*) is composed by two main modules, the *acquisition tool* and the *execution tool*. Both tools need to access data stored into a set of databases. In particular, the acquisition tool manages the representation of clinical guidelines, which are physically stored into a dedicated database, called CG DB. Moreover, it interacts with: the Pharmacological DB, storing a structured list of drugs and their costs; the Resource DB, listing the resources that are available in a given hospital (it is therefore used to represent the context-dependent version of a guideline); the ICD DB, containing an international coding system of diseases; the Clinical DB, providing a "standard" terminology to be used when building a new guideline, and storing the descriptions and the set of possible values of clinical findings. The interaction with the Clinical DB during acquisition allows for standardization (since experts are forced to use

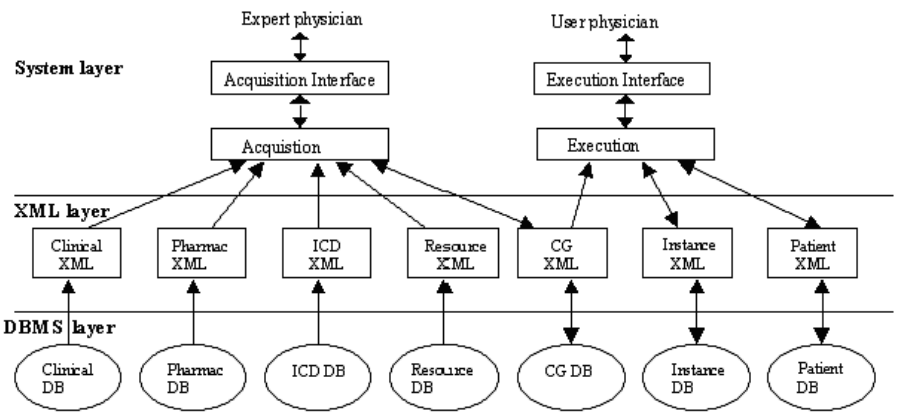



**Fig. 1.** GLARE's three-layered architecture

the same vocabulary) and for correctness (since only values for findings that are compatible with the range of values fixed in the Clinical DB itself can be specified).

The execution module executes a guideline for a specific patient, taking into account the patient's data (automatically retrieved from a database called Patient DB). This tool stores the status of the execution in the Instance DB and interacts with the user-physician via a user-friendly graphical interface.

The lowest layer of the architecture (*DBMS layer*) is made by the DBMS, that physically stores the different databases described above. However, in GLARE, the interaction between the acquisition and the execution tools with such databases is not a direct one, since it is mediated by the introduction of an intermediate layer (*XML layer*). The XML layer consists of a set of XML documents (one for each database). XML acts as an interlingua between the system layer and the DBMS layer: the acquisition and execution modules actually interact only with the XML layer, through which they obtain the knowledge stored into the DBMS. The use of XML as an interlingua allows us to express the guidelines in a format with characteristics of legibility, and to publish them on the Web, rendering easy their dissemination. On the other hand, the DBMS layer grants a homogenous management of the data, by integrating the guideline representation with the pre-existent Hospital Information System in the same physical DBMS.

The three-layered architecture makes GLARE independent of the commercial DBMS adopted by the particular hospital. In fact, the interaction between the DBMS and the XML layer is devoted to a single software module (a Java package). Changing the DBMS only requires to modify such module and these changes are quite limited and well-localized.

## 3 Representation Formalism

In order to guarantee usability of GLARE to user-physicians not expert in Computer Science, we have defined a *limited set* of clear representation primitives, covering most of the relevant aspects of a guideline. In particular, we have focused the attention on the concept of action, a basic primitive notion for describing clinical guidelines. We use the notion of "action" in quite a *broad sense*, in order to indicate the different activities which may characterize a diagnosis, or the application of a given therapy, or the finding/retrieving of information, or other clinical tasks. Given this notion, a guideline itself can be conceived as a complex action, composed by a number of elementary actions. We distinguish between *atomic* and *composite* actions. *Atomic actions* can be regarded as elementary steps in a guideline, in the sense that they do not need a further de-composition into sub-actions to be executed. *Composite actions* are composed by other actions (atomic or composite).

Four different types of *atomic actions* can be distinguished: *work actions*, *query actions*, *decisions* and *conclusions*. *Work actions* are atomic actions which must be executed at a given point of the guideline, and can be described in

terms of a set of attributes, such as name, (textual) description, cost, time, resources, goals. *Query actions* are requests of information, that can be obtained from the outside world (physicians, databases, knowledge bases). *Decision actions* are specific types of actions embodying the criteria which can be used to select from alternative paths in a guideline. In particular, *diagnostic decisions* are represented as an open set of triples ⟨*diagnosis*, *parameter*, *score*⟩ (where, in turn, a *parameter* is a triple ⟨*data*, *attribute*, *value*⟩), plus a threshold to be compared with the different diagnoses' scores. On the other hand, *therapeutic decisions* are based on a pre-defined set of parameters: effectiveness, cost, side-effects, compliance, duration. Finally, *conclusions* represent the explicit output of a decision process (for instance, assuming a given diagnostic hypothesis is a typical conclusion of a diagnostic decision action).

*Composite actions* are defined in terms of their components, via the has-part relation (this supports for top-down refinement in the description of guidelines). On the other hand, a set of *control relations* establishes which actions might be executed next and in what order. We distinguish among four different control relations: *sequence*, *controlled*[1], *alternative* and *repetition*.

A distinguishing feature of GLARE is its capability of representing (and treating) temporal constraints. Temporal constraints play a fundamental role in both the description and the execution of clinical guidelines. We have worked to design a temporal representation formalism as expressive as possible, still maintaining the tractability of the temporal reasoning process. Our formalism allows one to represent the (minimum and maximum) duration of each non-composite action. Temporal constraints can also be associated with control relations between actions. In the sequence and alternative relations, one can indicate the minimum and/or maximum delay between actions. In a controlled relation, one can specify the minimum and/or maximum distance between any pair of endpoints of the actions involved. On the basis of such distances, one can express both *qualitative* constraints between actions (however, only *continuous pointizable relations* can be coped with [23]) and *quantitative* ones. Finally, two different ways of specifying repetitions are defined (and can be combined): one can state that the action has to be performed until a given *exit condition* becomes true, or can specify duration (*frame time*) for the repetitions. In both cases, the *frequency* of the repetitions in time has to be specified as well; then, several other parameters must/can be provided.

**Ex.1** *For six months, perform action A twice each five days for twenty days, and then suspend for ten days.*

The *frame time* (henceforth called *FT* for short) can be defined as "*the interval which contains all the instances of the event*" [3], ("*for six months*" in Ex.1). The description of repeated periodic events splits FT into a sequence of intervals when actions are performed (called *action-times* - *AT*; "*twenty days*"

[1] Controlled relations are used to represent temporally constrained actions, such as "*A* during *B*", "start of *A* at least 1 hour after the beginning of *B*", and so on.

in Ex.1) and "pause" intervals (*delay time* - $DT$; "*ten days*" in Ex.1). In turn, $AT$s are split into *I-times* ($IT$; "*five days*" in Ex.1) where actions are actually performed (if $DT$ is null, $AT$ coincides with $IT$). Finally, we call the number of actions in each I-time "*frequency*" ($freq$; two in Ex.1).

Besides these "explicit" constraints, also the implicit constraints implied by the has-part relations between actions have to be taken into account [18].

## 4 Acquisition

The acquisition module is a user-friendly tool that provides expert physicians with:

(i) a graphical interface, which supports primitives for drawing the control information within the guideline, and ad hoc windows to acquire the internal properties of the objects;
(ii) facilities for browsing the guideline;
(iii) "intelligent" help and consistency checking (see next subsection).

### 4.1 Consistency Checking

The acquisition tool provides an "intelligent" interface supporting expert physicians in the acquisition of a guideline, relying on different forms of consistency checking.

*Name and range checking* is automatically triggered whenever the expert physician introduces a new term or value within the description of an action in a guideline, and forces her/him to use only terms/values that have already been defined within the Clinical DB. Whenever the expert physician introduces a node or arc, different controls are automatically activated to check whether the new element is consistent with several *logical design criteria*. For example, alternative arcs may only exit from a decision action. Finally, a "semantic" checking regards the *consistency of temporal constraints* in the guideline. This checking is automatically triggered whenever the expert physician saves a guideline. In fact, alternative sequences of actions and sub-actions may form graph structures, and the constraints on the minimum and maximum durations of actions and minimum and maximum delays between actions have to be propagated throughout the graph, to verify consistency.

While GLARE provides users with an *interface* high-level language to express temporal constraints, the temporal reasoning facility maintains a homogeneous *internal* representation of such constraints, on which the temporal reasoning algorithms operate. We based the design of the internal representation formalism on the "classical" *bounds on differences* approach and on the STP (Simple Temporal Problem) framework [1]. This framework takes into account conjunctions (sets) of bounds on the distance between pairs of time points (of the form $c \leq P1 - P2 \leq d$), and has very nice computational properties: correct and complete temporal reasoning (e.g., for consistency checking) can be performed in cubic

time by a classical all-to-all-shortest-paths algorithm (such as Floyd-Warshall's one), which also provides the *minimal network* of the temporal constraints [1].

Most of the temporal constraints provided by GLARE's interface formalism can be easily represented by the STP framework. Each action in a guideline (including composite actions) can be represented by its starting and its ending point. Thus, the duration of an action can be modeled as the distance between its endpoints. Delays are directly modeled as distances between points, as well as qualitative temporal constraints. Unfortunately, the STP framework must be significantly extended if one wishes to deal with *repetitions*. We propose to represent the constraints regarding repetitions into separate STP frameworks, one for each repeated action. Thus, in GLARE, the overall set of constraints in a guideline is represented by a *tree of STP frameworks* (*STP-tree* henceforth). The root of the tree is the STP which homogeneously represents the constraints between all the actions (composite and atomic) in the guideline, except repeated actions (which are composite actions, by our definition). Each node in the STP-tree is an STP, and has as many children as the number of repeated actions it contains. Each arc in the tree connects a pair of points in an STP (the starting and ending point of a repeated action) to the STP containing the constraints between the related subactions, and is labeled with the list of properties describing the temporal constraints on the repetitions ($AT$, $DT$ etc.; see Ex.2 below). Figure 2 shows the STP-tree representing the temporal constraints in Ex.2.

**Ex.2** *One possible therapy for multiple mieloma is made by six cycles of 5-day treatment, each one followed by a delay of* 23 *days (for a total* $FT$ *of* 24 *weeks, divided into six repetitions of an* $AT$ *of* 5 *days, followed by a* $DT$ *of* 23 *days. The overall therapy is reported as the root of the STP-tree in figure 2). Within each 5-day cycle,* 2 *inner cycles can be distinguished: the melphalan treatment, to be provided twice a day (*$AT$*=*$IT$*), for each of the* 5 *days (*$FT$*), and the prednisone treatment, to be provided once a day, for each of the* 5 *days. These two treatments must be performed in parallel (see the temporal constraints in node* $N2$ *in figure 2), and are shown as leaves of the STP-tree (nodes* $N3$ *and* $N4$ *respectively).*

Temporal consistency checking proceeds in a top-down fashion, starting from the root node of the STP-tree. The root is a "standard" STP, so that Floyd-Warshall's algorithm can be applied. Then, we proceed towards the leaves of the tree. For each node in the tree other than the root, we apply $ALGO1$ (see [18] for more details):

$ALGO1$**: temporal consistency of guidelines**

1. the consistency of the constraints used to specify the repetition taken in isolation is checked;
2. the "extra" temporal constraints regarding the repetition are mapped onto bounds on difference constraints;
3. Floyd-Warshall's algorithm is applied to the constraints in the STP plus the "extra" bounds on difference constraints determined at step 2.

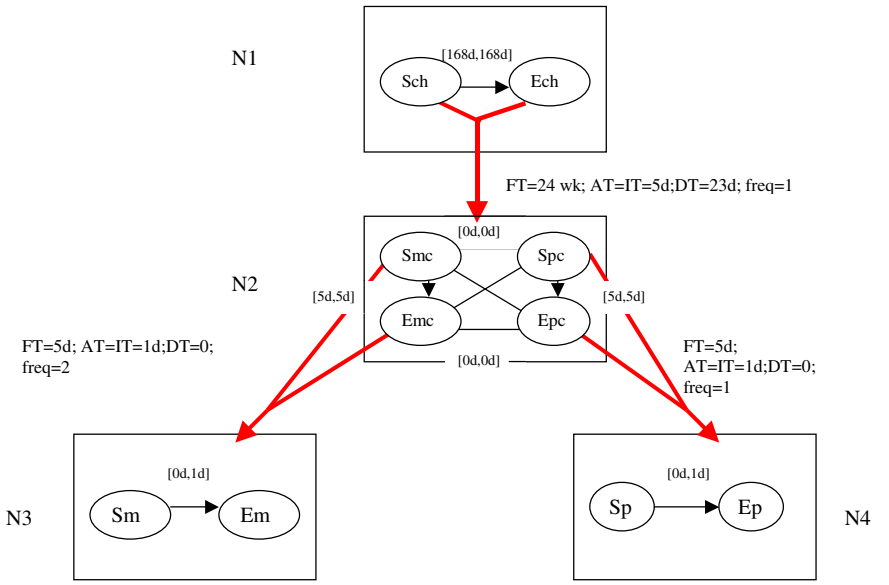



**Fig. 2.** STP-tree for the multiple mieloma chemotherapy guideline. Arcs between nodes in a STP are labeled by a pair $[n, m]$ representing the minimum and maximum distance between them. Arcs from a pair of nodes to a child STP represent repetitions

**Property 1**. *ALGO*1 is correct, complete, and tractable (since it operates in $O(N^3)$, where $N$ is the number of actions in the guideline).

## 5 Execution

The typical use of our execution tool is "on-line": a user physician executes a guideline applied to a specific patient (i.e., s/he instantiates a general guideline considering the data of a given patient). However, we also envision the possibility of adopting our execution tool for "off-line" execution (this might be useful in different tasks, including education, critiquing and evaluation). In both cases, temporal reasoning and decision support facilities may be resorted to (see next subsections).

### 5.1 Temporal Reasoning Facilities

The execution tool exploits temporal consistency checking as well. Each action in a guideline represents a *class* (set) of instances of actions, in the sense that it will have specific instantiations for specific executions of the guideline itself. When a guideline is executed on a specific patient, specific *instances* of such actions are performed at specific times. We suppose that the exact times of all the actions

in the guideline which have been executed are given in input to our system. Thus, we have to check that they respect (i.e., are consistent with) the temporal constraints they inherit from the classes in the general guideline. Moreover, also the (implicit) temporal constraints conveyed by the has-part relations between actions in the guideline must be respected, as well as those involved by periodicity and repetitions.

In a broad sense, periodic events are special kinds of classes of events, i.e., classes whose instances must respect a periodic temporal pattern. However, while inheritance of constraints about duration, delays and ordering regards single instances (duration) or pairs of instances (delays, precedence), periodicity constraints concern whole sets of instances, imposing constraints on their cardinality and on the temporal pattern they have to respect. Finally, notice that the interplay between part-of relations and periodic events might be quite complex to represent and manage. In fact, in the case of a composite periodic action, the temporal pattern regards the components, which may be, recursively, composite and/or periodic actions (see Ex.2).

Finally, notice that, when considering instances, one should also take into account the fact that guidelines have a "predictive" role. E.g., if one has observed a given action $E1$ which is an instance of a class of actions $E$ in a guideline, and the class $E'$ follows $E$ in the guideline itself, one expects to observe an instance of $E'$ in a time consistent with the temporal constraints between the classes $E$ and $E'$. We assume that, as regards the treatment of hospitalized patients, we have *complete observability*, i.e., that each execution of an action of the guideline is reported in the clinical record of the patient, together with its time of occurrence. Thus the consistency checking must consider "prediction", since not having observed an instance of an action indicates an inconsistency, unless the temporal constraints impose that it may also be executed in a time after NOW.

Our temporal reasoning algorithm can be schematized as follows:

*ALGO*2**: temporal consistency on guidelines' execution**

1. the existence of non-observed instances whose occurrence is predicted by the guideline is hypothesized;
2. all the constraints in the general guidelines are inherited by the corresponding instances (considering both observed and hypothesized instances). This step also involves "non-standard" inheritance of constraints about periodicity;
3. constraint propagation is performed on the resulting set of constraints on instances (via Floyd-Warshall's algorithm), to check the consistency of the given and the inherited constraints;
4. if constraints at step 3 are consistent, it is further checked that such constraints do not imply that any of the "hypothesized" instances should have started before NOW.

**Property 2**. Our consistency checking algorithm *ALGO*2 is correct, complete, and tractable (since it operates in $O((N + M)^3)$, where $N$ is the number of

actions in the guideline and $M$ the number of instances of actions which have been executed.

A detailed analysis of our temporal reasoning algorithm, and of Property 2 is outside the goals of this paper, and can be found in [17].

### 5.2 Hypothetical Reasoning Facility

GLARE's execution tool also incorporates a decision support facility (called *hypothetical reasoning*), able to assist physicians in choosing among different therapeutic or diagnostic alternatives. The default execution of decision actions works as follows. As regards *diagnostic decisions*, the execution module automatically retrieves the parameter values from the Patient DB, evaluates the scores for every alternative diagnosis, and then compares them with the corresponding threshold. All alternative diagnoses are then shown to the user-physician, together with their scores and the threshold, and the tool lets the user choose among them (a warning is given if the user chooses a diagnosis which does not exceed the threshold). The execution of a *therapeutic decision* simply consists in presenting the effectiveness, cost, side-effects, compliance, and duration of each alternative to the physician, thus allowing her/him to select one of them. On the other hand, through the adoption of the hypothetical reasoning facility, it is possible to compare different paths in the guideline, by simulating what could happen if a certain choice was made. In particular, users are helped in gathering various types of information, needed to discriminate among alternatives. As a matter of fact, in many cases, therapeutic and/or diagnostic decisions should not be taken on the basis of "local information" alone, i.e. by considering just the decision criteria associated with the specific decision action at hand, but one should also take into account information stemming from relevant alternative paths. In particular, the resources needed to perform all the actions found along each alternative path (starting from the decision at hand), the costs and the times required to complete them, are meaningful selection parameters. The unique feature of this tool is its capability of retrieving such "global information". This facility can be used both in the on-line and in the off-line execution mode.

Technically speaking, to provide a projection of what could happen in the rest of the guideline in case the user selected a given alternative, the tool works as follows. Through the execution tool graphical interface, the physician is asked to indicate on the graph the starting node (normally the decision at hand) of the paths to be compared and (optionally) the ending nodes (otherwise all possible paths exiting the starting node will be taken into consideration). Relevant decision parameters (costs, resources, times) will be gathered from the selected portions of the guideline in a semi-automatic way. In particular, whenever a decision action is reached within each path, the user is allowed to choose a subset of alternatives, by checking the corresponding buttons in a pop up window. For a diagnostic decision, s/he may want to allow all alternatives to be considered, or s/he could limit the search to the diagnoses that obtained a score exceeding the threshold, or to a subset of these diagnoses themselves. When dealing with

a therapeutic action, again the user could allow all alternatives to be evaluated, or could mark the therapies s/he expects to be equivalent for the patient under examination, or a subset of them. Making a restriction means that, on the physician's opinion, the other paths are not interesting for comparison, and they will be ignored by the hypothetical reasoning process. If a composite action is found, it is expanded in its components, and the hypothetical reasoning facility is recursively applied to each of them, by analyzing all the decision actions that appear at the various decomposition levels. At the end of this process, the tool displays the values of the collected parameters for each one of the selected paths. The final decision is then left to the physician. Note that while resources in a path are simply listed, and costs are summed up (in the case that an exit condition is specified, the cost of each iteration will be calculated), the temporal constraint propagation techniques discussed so far are necessary in order to deal with the temporal parameters.

## 6 Testing

We have already tested our prototype acquisition and representation system considering different domains, including bladder cancer, reflux esophagitis and heart failure. In the case of bladder cancer, the expert physicians started designing the guideline algorithm from scratch, directly using our acquisition tool (after a brief training session), and exploiting the facilities (e.g., consistency checking) it provides.

In the cases of reflux esophagitis and heart failure, the physicians started with guideline algorithms previously described on paper (using drawings and text), and used our acquisition tool to introduce them into a computer format. The acquisition of a clinical guideline using our system was reasonably fast (e.g., the acquisition of the guideline on heart failure required 3 days).

In all the tests, our representation formalism (and the acquisition tool) proved to be expressive enough to cover the clinical algorithms.

## 7 Comparisons and Conclusions

In this paper, we highlighted the most innovative features of GLARE, a domain - independent framework to acquire, represent and execute clinical guidelines. In the latest years, many approaches agreed that providing a semi-automatic treatment of clinical guidelines is very advantageous, and that AI techniques can be fruitfully applied to achieve such a goal.

Among the approaches in the literature, we think that PROforma [4] and Asbru [15] are the closest ones to GLARE. However, two distinguishing features of the GLARE approach, that clearly highlight the advantages of applying AI techniques to clinical guideline tools, can be outlined:

(i) GLARE provides "intelligent" mechanisms for consistency checking (see however [13], where the correctness and completeness of the activation conditions of subtasks are automatically checked). Specific attention is devoted

to the treatment of temporal constraints between atomic, periodic and/or repeated actions in both the acquisition (section 4) and in the execution (section 5) phases;

(ii) GLARE provides user physicians with the hypothetical reasoning facility, a practical way of comparing alternative paths in a guideline on the basis of a chosen set of parameters (see section 5).

In particular, (i) and (ii) are also the innovative features of the approach we described in this paper (together with the introduction of the "intermediate" XML layer in the architecture; see section 2) with respect to our initial approach, as described in [20].

More generally, GLARE, as well as PROforma, Asbru, and many other approaches, shows that the adoption of AI techniques can provide relevant advantages in the (semi-)automatic treatment of clinical guidelines, especially regarding the user - friendliness of the tools being built. In turn, user-friendliness seems to be one of the most crucial aspects in the dissemination and actual adoption of computer science tools within the medical community.

## References

[1] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. In R. J. Brachman, H. J. Levesque, and R. Reiter, editors, *Knowledge Representation*, pages 61–95. MIT Press, London, 1991. 540, 541

[2] D. B. Fridsma (Guest Ed.). Special issue on workflow management and clinical guidelines. *Journal of the American Medical Informatics Association*, 1(22):1–80, 2001. 537

[3] F. Van Eynde. Iteration, habituality and verb form semantics. In *Proc. of third Conference of European Chapter of the Association for Computational Linguistics*, pages 270–277, 1987. 539

[4] J. Fox, N. Johns, A. Rahmanzadeh, and R. Thomson. Disseminating medical knowledge: the proforma approach. *Artificial Intelligence in Medicine*, 14(1-2):157–182, 1998. 537, 545

[5] C. Gordon. Practice guidelines and healthcare telematics: towards an alliance. In C. Gordon and J. P. Christensen, editors, *Health Telematics for Clinical Guidelines and Protocols*, pages 3–15. IOS Press, Amsterdam, 1995. 536

[6] C. Gordon and J. P. Christensen, editors. *Health Telematics for Clinical Guidelines and Protocols*. IOS Press, Amsterdam, 1995. 537

[7] J. M. Grimshaw and I. T. Russel. Effect of clinical guidelines on medical practice: a systematic review of rigorous evaluation. *Lancet*, 342:1317–1322, 1993. 536

[8] G. Molino. From clinical guidelines to decision support. In *Proc. Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making, LNAI 1620*, pages 3–12, 1999. 536

[9] M. A. Musen, C. W. Carlson, L. M. Fagan, S. C. Deresinski, and E. H. Shortliffe. Automated support for community-based clinical research. In *Proc. 16th Annual Symposium on Computer Applications in Medical Care*, pages 719–723, 1992. 537

[10] M. A. Musen, S. W. Tu, A. K. Das, and Y. Shahar. Eon: a component-based approach to automation of protocol-directed therapy. *Journal of the American Medical Informatics Association*, 6(3):367–388, 1996. 536

[11] M. Peleg and et al A. A. Boxawala. Glif3: The evolution of a guideline representation format. In *Proc. AMIA'00*, pages 645–649, 2000. 537

[12] I. Purves. Computerised guidelines in primary health care: Reflections and implications. In C. Gordon and J. P. Christensen, editors, *Health Telematics for Clinical Guidelines and Protocols*, pages 57–74. IOS Press, Amsterdam, 1995. 536

[13] S. Quaglini, L. Dazzi, L. Gatti, M. Stefanelli, C. Fassino, and C. Tondini. Supporting tools for guideline development and dissemination. *Artificial Intelligence in Medicine*, 2(14):119–137, 1998. 545

[14] S. Quaglini, M. Stefanelli, A. Cavallini, G. Miceli, C. Fassino, and C. Mossa. Guideline-based careflow systems. *Artificial Intelligence in Medicine*, 1(20):5–22, 2000. 537

[15] Y. Shahar, S. Mirksch, and P. Johnson. The asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines. *Artificial Intelligence in Medicine*, 14(1-2):29–51, 1998. 536, 545

[16] R. N. Shiffman, B. T. Karras, A. Agrawal, R. Chen, L. Menco, and S. Nath. Gem: a proposal for a more comprehensive guideline document model using xml. *Journal of the American Medical Informatics Association*, 5(7):488–498, 2000. 536

[17] P. Terenziani and L. Anselma. Towards a temporal reasoning approach dealing with instance-of, part-of, and periodicity. Accepted for publication in Proc. TIME'03. IEEE Press, 2003. 544

[18] P. Terenziani, C. Carlini, and S. Montani. Towards a comprehensive treatment of temporal constraints in clinical guidelines. In *Proc. TIME'02*, pages 20–27. IEEE Press, 2002. 540, 541

[19] P. Terenziani, F. Mastromonaco, G. Molino, and M. Torchio. Executing clinical guidelines: temporal issues. In *Proc. AMIA'00*, pages 848–852, 2000. 537

[20] P. Terenziani, G. Molino, and M. Torchio. A modular approach for representing and executing clinical guidelines. *Artificial Intelligence in Medicine*, 23(3):249–276, 2001. 537, 546

[21] P. Terenziani, S. Montani, A. Bottrighi, G. Molino, and M. Torchio. Supporting physicians in taking decisions in clinical guidelines: the glare's "what if" facility. *Journal of the American Medical Informatics Association, Proc. Annual Fall Symposium*, 2002. 537

[22] S. W. Tu, M. G. Kahn, M. G. Musen, J. K. Ferguson, E. H. Shortliffe, and L. M. Fagan. Episodic skeletal-plan refinement on temporal data. *Comm. ACM*, 32(12):1439–1455, 1989. 537

[23] L. Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 1(7):4–28, 1994. 539

# Two Paradigms for Natural-Language Processing

Robert C. Moore

Microsoft Research, Usa

In the history of artificial intelligence, many controversies have divided the field. The 1970s saw the clash of "logical" AI vs. "procedural" AI, which then became generalized into "neat" AI vs. "scruffy" AI. In the 1980s and 1990s, "connectionist" and "reactive" approaches to AI arose to challenge the predominant paradigm based on reasoning with explicit representations. Perhaps due to the increasing maturity of the field (or its practitioners), such basic disagreements rarely seem to generate as much heated argument as in prior years. Nevertheless, divisions exist today in AI that are at least as fundamental as any of those mentioned above.

In my view, the most significant such split in the field today concerns the question of how knowledge is to be acquired by intelligent systems. The two principal paradigms can be described as knowledge engineering and data-driven learning. The term "knowledge engineering" originated in connection with a particular approach to building expert systems, but I will use it in the broader sense of any AI system dependent on human experts entering large amounts of knowledge. In contrast, data-driven learning generally involves developing some sort of abstract (often statistical) model of a problem and training the model on large amounts of data, either as it naturally occurs or with human annotation.

No subfield of AI has been more profoundly affected by the contrast between these two approaches to knowledge acquisition than natural-language processing. Over the last fifteen years or so, NLP has gone from being almost completely in the knowledge-engineering camp to predominantly focused on data-driven-learning-based approaches. In this talk, we will review the advantages and disadvantages of both of these paradigms for NLP, looking at some of the major ideas and achievements of each. We will conclude by examining the prospects for a synthesis that combines the benefits of both.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, p. 548, 2003.
© Springer-Verlag Berlin Heidelberg 2003

# Robotics and AI: From Intelligent Robots to Neuro-robotics

Paolo Dario

Scuola Superiore Sant'Anna, Pisa, Italy

In the last few years robotics has increasingly been recognized and accepted not only as a field for application in industry and services, but also as a potentially ideal application domain for Artificial Intelligence. AI has contributed significantly to the progress of various areas of robotics, particularly those of perception, sensory-motor coordination and intelligent behavior.

AI research, in fact, has focused on developing entities with intelligence comparable to that of humans-that is, with the capability of reasoning and managing knowledge. With some controversy, eventually AI research has recognized (as Rodney Brooks at MIT first proposed) that interaction with the physical world is critical for developing intelligence. Since then, many research groups have tried to develop humanoid "bodies". According to this approach, human-like intelligence relates not only to reasoning but also to learning, perceiving, and interpreting the physical world and to interacting with the world and humans.

These goals are much more difficult to implement in machines than pure reasoning. As Tommaso Poggio pointed out, humans have developed reasoning rules only in the last few millennia; perception seems so natural to us because nature has refined it over millions of years. In fact, AI has recently achieved what was considered its biggest challenge for decades-defeating a human champion in the game of chess. AI's new, much more challenging goal is now to develop humanoid robots that can play, and possibly win, against a human soccer team. If this concept of 'embodiment' proposes robotics as a tool for developing artificial intelligence, recent technological advances in the development of biomimetic robotic artefacts and of their sensory-motor and behavioral schemes led to a further conceptual advancement: the application of robotics in understanding intelligence itself. This area is sometimes referred to as "neuro-robotics", a term that underlines the importance of the contribution of neuroscience and robotics. Neuro-robots are built by implementing models formulated by neuroscientists, in order to serve as experimental platforms for the validation of such models.

Even though still in its infancy, this emerging field is very promising. In this talk, a number of experimental projects will be presented and discussed. The ultimate goal of these projects is to assess the real contribution that the field of neuro-robotics can bring to the comprehension of the human brain and its functions, and to identify the conditions for the exploitation of the potential of this approach.

A. Cappelli and F. Turini (Eds.): AI*IA 2003, LNAI 2829, p. 549, 2003.
© Springer-Verlag Berlin Heidelberg 2003