Lecture Notes in Artificial Intelligence 2522

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Tokyo

Troels Andreasen
Amihai Motro
Henning Christiansen
Henrik Legind Larsen (Eds.)

Flexible Query Answering Systems

5th International Conference, FQAS 2002
Copenhagen, Denmark, October 27-29, 2002
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Authors

Troels Andreasen
Henning Christiansen
Roskilde University
Building 42-1, P.O. Box 260, 4000 Roskilde, Denmark
E-mail: {troels,henning@ruc.dk}

Amihai Motro
Department of Information and Software Engineering
School of Information Technology and Engineering
George Mason University, Fairfax, Virginia 22030-4444, USA
E-mail: ami@gmu.edu

Henrik Legind Larsen
Dept. of Computer Science and Engineering, Aalborg University Esbjerg
Niels Bohrs Vej 8, 6700 Esbjerg, Denmark
E-mail: legin@cs.aue.auc.dk

Cataloging-in-Publication Data applied for

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at http://dnb.ddb.de

CR Subject Classification (1998): H.3, I.2, H.2, H.4, H.5

ISSN 0302-9743
ISBN 3-540-00074-7 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna e. K.
Printed on acid-free paper SPIN: 10873099 06/3142 5 4 3 2 1 0

Preface

This volume constitutes the proceedings of the Fifth International Conference on Flexible Query Answering Systems, FQAS 2002, held in Copenhagen, Denmark on October 27–29, 2002.

FQAS is the premier conference for researchers and practitioners concerned with the vital task of providing easy, flexible, and intuitive access to information for every type of need. This multidisciplinary conference draws on several research areas, including databases, information retrieval, knowledge representation, soft computing, multimedia, and human–computer interaction. Previous FQAS events were held in 1994, 1996, 1998, and 2000.

The overall theme of the FQAS conferences is innovative query systems aimed at providing easy, flexible, and intuitive access to information. Such systems are intended to facilitate retrieval from information repositories such as databases, libraries, and the World-Wide Web. These repositories are typically equipped with standard query systems which are often inadequate, and the focus of FQAS is the development of query systems that are more expressive, informative, cooperative, and productive.

These proceedings contain 29 original papers that relate to the topic of users posing queries and systems producing answers. The papers cover the fields: Database Management, Information Retrieval, Domain Modeling, Knowledge Representation and Ontologies, Knowledge Discovery and Data Mining, Artificial Intelligence, Classical and Non-classical Logics, Computational Linguistics and Natural Language Processing, Multimedia Information Systems, and Human–Computer Interaction.

We wish to thank the contributors for their excellent papers and the referees, publisher, and sponsors for their effort. Special thanks also to the invited speakers, members of the Advisory Board, and members of the Program Committee. They made the success of FQAS 2002 possible.

September 2002

Troels Andreasen
Amihai Motro
Henning Christiansen
Henrik Legind Larsen

Organization

FQAS'2002 is organized by the Department of Computer Science, Roskilde University, the Department of Information and Software Engineering, George Mason University, and the Department of Computational Linguistics, Copenhagen Business School.

Organizing Committee

Conference Co-Chairs:	Troels Andreasen, Roskilde University, Denmark
	Henning Christiansen, Roskilde University, Denmark
Program Co-Chairs:	Amihai Motro, George Mason University, USA
	Troels Andreasen, Roskilde University, Denmark
Local Chair:	Hanne E. Thomsen,
	Copenhagen Business School, Denmark

Program Committee

T. Andreasen, Denmark
L. Bertossi, Canada
G. Bordogna, Italy
B. Bouchon-Meunier, France
T. Brauner, Denmark
L. Cholvy, France
J. Chomicki, USA
H. Christiansen, Denmark
F. Crestani, UK
J. Carlos Cubero, Spain
E. Damiani, Italy
R. De Caluwe, Belgium
G. De Tre, Belgium
R. Demolombe, France
M. Detyniecki, France
D. Dubois, France
C. Enguehard, France
R. Felix, Germany
J. Fischer Nilsson, Denmark
N. Fuhr, Germany
R. George, USA
P. Ingwersen, Denmark
A. Helms Jorgensen, Denmark
J. Kacprzyk, Poland
J. Karlgren, Sweden
N. Kasabov, New Zealand
E. Kerre, Belgium
R. Knappe, Denmark
L.T. Koczy, Hungary
D. Kraft, USA
W. Kurutach, Thailand
M. Lalmas, UK
H.L. Larsen, Denmark
A. Mark Pejtersen, Denmark
C. Marsala, France
M. Jose Martin-Bautista, Spain
S. Miyamoto, Japan
A. Morris, USA
A. Motro, USA
N. Mouaddib, France
F. Luis Neves, Portugal
V. Novak, Czech Republic
P. Paggio, Denmark
F. Petry, USA
O. Pivert, France
O. Pons, Spain
Z. Ras, USA
G. Raschia, France

Table of Contents

Context – Sensitive Query Expansion Based on Fuzzy Clustering of Index Terms

Giorgos Akrivas, Manolis Wallace, Giorgos Stamou, and Stefanos Kollias

Image, Video and Multimedia Laboratory,
Department of Electrical and Computer Engineering,
National Technical University of Athens,
15773 Zografou, Greece
`{gakrivas,wallace}@image.ntua.gr`, `{gstam, stefanos}@softlab.ntua.gr`
`http://www.image.ntua.gr`

Abstract. Modern Information Retrieval Systems match the terms contained in a user's query with available documents through the use of an index. In this work, we propose a method for expanding the query with its associated terms, in order to increase the system recall. The proposed method is based on a novel fuzzy clustering of the index terms, using their common occurrence in documents as clustering criterion. The clusters which are relevant to the terms of the query form the query context. The terms of the clusters that belong to the context are used to expand the query. Clusters participate in the expansion according to their degree of relevance to the query. Precision of the result is thus improved. This statistical approach for query expansion is useful when no a priori semantic knowledge is available.

1 Introduction

An *Information Retrieval System (IRS)* consists of a database, containing a number of documents, an *index*, that associates each document to its related terms, and a *matching mechanism*, that maps the user's query (which consists of *terms*), to a set of contained documents. Quite often, the user's query and the index are fuzzy, meaning that the user can somehow supply the degree of importance for each term, and that the set of associated terms for each document also contains degrees of association. In this case, the returned documents are sorted, with the one that best matches the user's query returned first [1].

It is possible that a query does not match a given index entry, although the document that corresponds to the entry is relevant to the query. For example, a synonym of a term found in a document may be used in the query. This problem, which is known as *word mismatch* [2], has been dealt with in the past with various methods [3], many of which have an iterative [4],[5],[6] or an interactive [7] manner. Such approaches rely on relevance feedback.

Still, the dominating approach to the problem of word mismatch is the use of a thesaurus containing, for each term, the set of its related ones. The process of enlarging the user's query with the associated terms is called *query expansion*;

it is based on the associative relation A of the thesaurus, which relates terms based on their probability to co-exist in a document [8],[9]. In such approaches, as is for example the LCA method described in [10], the thesauri are generally created based on term co-occurrences in documents. Unfortunately, as has been shown on [11], the simple use of a general thesaurus for query expansion provides limited improvement. This can be justified as follows: query expansion by using a thesaurus results in the selection of numerous documents that are related to the query terms only if taken out of context.

In order to make query expansion sensitive to context, the matching mechanism must be made more intelligent. Usually, this involves a *semantic encyclopedia*, which can be used as a means to provide semantics to the user's query [12],[13],[14]. The semantics are used to extract the query context, which is subsequently used to direct query expansion towards terms that are related to the context. Such a method has been proposed by the authors in [16].

However, a semantic encyclopedia is not always available, because it requires tedious and time-consuming labor by a human expert. Moreover, the scope of the encyclopedia might be unrelated to some of the terms. In this work, we propose a method that extracts the query context using the index of the IRS, instead of a knowledge base. The method is as follows: A fuzzy clustering of the index terms, based on statistical measurements on the index, extracts a set of possible query contexts. These clusters are compared with the query and the context of the query is expressed on terms of clusters. The extracted context is subsequently used to direct query expansion towards the clusters that are related to it.

The paper is organized as follows: In section 2, we provide the mathematical framework. In section 3 we provide the algorithm for fuzzy clustering of index terms. In section 4.1 we use the clusters of terms to detect the context of the query. In sections 4.2 and 4.3, we use context to expand the user's query in a meaningful way. Finally, sections 5 and 6 provide a simulation example and our final conclusions, respectively.

2 Mathematical Framework

Before continuing, we provide the reader with the mathematical framework on fuzzy sets and relations.

Let $S = \{s_1, s_2, \ldots, s_n\}$, where $n = |S|$, denote the set of indexed terms.

A *fuzzy set* q on S is a function $q : S \to [0, 1]$. Its *scalar cardinality* is defined as $|q| = \sum_{s \in S} q(s)$. From now on, for the fuzzy set q, we will use the sum notation $q = \sum_{i \in \mathbb{N}_n} s_i/q_i$, where $q_i = q(s_i)$.

The subset relation, for fuzzy sets, is defined as $q_1 \subseteq q_2 \Leftrightarrow q_1(s) \leq q_2(s), \forall s$

A *fuzzy binary relation* on $X \times Y$ is a fuzzy set on $[0, 1]$, i.e. a function $R : X \times Y \to [0, 1]$.

The *intersection* and *union* of two fuzzy relations P and Q defined on the same set S are defined as:

$[P \cap Q](x, y) = t(P(x, y), Q(x, y))$

$[P \cup Q](x, y) = u(P(x, y), Q(x, y))$

where t and u are a triangular norm and a triangular co-norm, respectively. The *standard* t-norm and t-conorm are, respectively, the min and max functions. In this work, we use the standard functions, except if noted otherwise.

The *power set* $\mathcal{P}(S)$ of S is the set of all subsets of S. $\mathcal{P}(S)$ contains 2^n members. The subset relation $\subset$ is a strict ordering relation on $\mathcal{P}(S)$.

3 Fuzzy Clustering of Indexed Terms

In this section, we propose a method for clustering of the indexed terms. Our purpose is to create clusters of terms that are highly associated with each other. The basic principle of the method is that a set of terms is considered to be a valid cluster when they are frequently encountered in the same documents. Obviously, the frequency of term co-occurrence in documents varies and consequently their association is a matter of degree; we will use the term *cluster validity* to express this degree.

According to the above, a cluster is described by the set of documents that correspond to its terms; however, a cluster term may be related to more documents than those that characterize the cluster. Terms that are contained in substantially more documents than those that characterize a cluster, must not be considered to be highly associated with it. In other words, the degree of membership a term to a cluster depends on the degree to which the documents that contain a term are limited to the cluster's characterizing documents.

Moreover, a term may appear in more than one group of documents and thus it may participate in more than one clusters. Therefore, clusters may overlap.

In the following, we will describe how the clusters can be created automatically. This process is based on an extensive indexing of a sufficient and representative set of documents of an archive; extensive indexing allows us to infer that term co-occurrence originates from their semantics. In other words, we interpret such a co-occurrence as a semantic association, which we may apply when considering the remaining documents. The degree of association depends on both cluster validity and the terms' degree of participation to the cluster.

Comparing the effort required to create the extensive index, with the one required to create a semantic encyclopedia, we can observe that indexing requires less effort because i) the thesaurus relations need not be supplied by the expert and ii) having a semantic encyclopedia does not relieve the expert from the need to index the documents.

Before continuing, we provide a few details on the index.

3.1 The Index of the Information Retrieval System

The index D is a relation that maps the set of terms S to the set of documents T. Although the index is in most cases crisp, we make the more general assumption that fuzziness is allowed. Thus, some terms are indexed to lesser degree than others. Therefore, D is a fuzzy relation $D : S \times T \to [0, 1]$.

By $D(s)$, we will denote the set of documents that contain the term s, i.e. the respective row of the index.

3.2 Structure in Documents and Proximity of Term Occurrence

Quite often, documents contain structure, in the form of subdocuments, sub-subdocuments and so on. In this case, terms that belong to the same part of the document are considered more associated than terms that belong to different parts. In this subsection, we consider how this affects the process of indexing.

Let as suppose that a document $t \in T$ contains two subdocuments, t_1, t_2. The two subdocuments are considered members of the set of documents T. Moreover, let us suppose that subdocument t_1 contains term s. Then, $D(s,t_1) = 1$ and $D(s,t_2) = 0$. The term s will be considered belonging to t to a lesser degree than 1, since it actually belongs to only one of its subdocuments. A way to estimate the importance of s in t is to use the percentage of the length of t_1 in the length of t. Thus, if t_1 is long and t_2 is short, then $D(s,t)$ will be close to 1 and vice versa.

Using the above principle, terms that belong to the same parts will have a greater degree of co-occurrence in documents than terms that just belong to the same document.

3.3 Detection of Clusters

Let $\mathcal{P}(S) = \{S_1, S_2, \dots, S_m\}$, where $m = 2^n$, be the power set of S. Each member $S_i = \{s_{ij} : j \in \mathbb{N}_{n_i}\}, n_i = |S_i|$ is a candidate cluster of terms. When we also consider degrees of association of terms to the cluster, the corresponding fuzzy cluster is:

$$S_i = \sum_{j \in \mathbb{N}_{n_i}} s_{ij}/w_{ij}$$

From now on, the term "cluster" will refer to both the crisp set and its fuzzy counterpart.

A cluster is considered valid with degree one if its members occur in the same documents; based on the assumption that the set of documents is representative, we use scalar cardinality as a measure of the degree of co-occurrence. Thus, validity shall be proportional to the number of documents that contain all terms, and inversely proportional to the number of documents that contain at least one term. Therefore, we define the validity of cluster S_i as:

$$w_i = \frac{\left| \bigcap_{s \in S_i} D(s) \right|}{\left| \bigcup_{s \in S_i} D(s) \right|}$$

In cluster S_i, term s_{ij} participates with degree one when the documents that contain it are the documents that contain all members of the cluster, i.e. when $D^*(S_i) = D(s_{ij})$. Similarly to the previous definition, we define w_{ij} as:

$$w_{ij} = \frac{\left|\bigcap_{s \in S_i} D(s)\right|}{|D(s_{ij})|}$$

Obviously, $w_{ij} \geq w_i$. Moreover, we assume that each term belongs to at least one document; therefore, the denominator in the definitions of w_{ij} and w_i will never be zero.

By defining an appropriate lower threshold c_v, and eliminating clusters S_i, for which $w_i < c_v$, we avoid non-meaningful clusters. We will use the term *validity criterion* for c_v.

However, there still may be superfluous clusters. In particular, two valid clusters $S_1 \subset S_2$ may be characterized by the same documents. In this case, S_1 is unnecessary and should be eliminated. To enable this, we define the Inclusion relation I, which is a fuzzy ordering relation on the set of clusters, i.e. $I : [\mathcal{P}(S)]^2 \to [0,1]$. I is a subset of the subset relation defined on the clusters. Therefore, $S_i \subset S_j \implies I(S_i, S_j) = 0, \forall S_i, S_j \in \mathcal{P}(S)$. Moreover, $I(S_i, S_j)$ approaches one as w_i approaches w_j.

Based on these conditions, we define the Inclusion relation as follows:

$$S_i \supseteq S_j \implies I(S_i, S_j) = \frac{w_i}{w_j}$$

If $I(S_i, S_j)$ falls above an appropriate threshold c_m, cluster S_j will be eliminated, as redundant. We will use the term *merging criterion* for c_m.

3.4 The Clustering Algorithm

Let us now proceed with the details of the clustering algorithm. We initialize the algorithm with the singletons $\{S_{11} = \{s_1\}, S_{12} = \{s_2\}, \dots, S_{1n_1} = \{s_{n_1}\}\}$, where $n_1 = n$.

The algorithm executes in n steps, and each step i uses as input clusters $\{S_{ij} : j \in \mathbb{N}_{n_i}\}$, with cardinality $|S_{ij}| = i$ and produces as output clusters $\{S_{i+1,j} : j \in \mathbb{N}_{n_{i+1}}\}$, with cardinality $|S_{i+1,j}| = i + 1$. Each step i executes as following:

- For each cluster $S_{ij}, j = 1, \dots, n_i - 1$:
  - For each cluster $S_{ik}, k = j + 1, \dots, n_i$:
    1. Compute the union $S_{ij} \cup S_{ik}$
    2. If it already exists, is invalid or has a cardinality different than $i+1$, then it is deleted
- Delete clusters that satisfy the merging criterion

The reason we choose to limit the output of step i to clusters of cardinality $i + 1$ is to ensure that the same clusters are not produced by different steps of the algorithm. Furthermore, as they will be computed again in consequent steps, this causes no loss.

By arranging clusters in a lexicographical order, we may avoid superfluous unions, as well checking for duplicate clusters and for clusters with invalid cardinality in step 2.

Both of the above contribute to the substantial reduction of the needed operations. By defining proper validity and merging criteria, and eliminating clusters that don't satisfy them, it is expected that the remaining clusters are both meaningful and non-redundant.

4 Context – Sensitive Query Expansion

As mentioned in section 1, query expansion enriches the query in order to increase the recall of the system. The presence of several terms in the query defines a context, which we use, in this section, to limit the expansion process, in order to improve the precision of the retrieval.

4.1 Detection of Context

As described in section 1, a query q is a fuzzy set defined on S. This means that any term $s_i \in S, i \in \mathbb{N}_n$ belongs to q to some degree $q_i = q(s)$. Of course, for most terms this degree is expected to be zero. Nevertheless, we assume that $q_i = 1$ for at least one term (i.e. q is normal, which means that the height is one). In this subsection, we express the context of the query in terms of the clusters that include it.

Let as first consider the case where the query is crisp. In this case $q = \{s_i\} \in \mathcal{P}(S)$. The Inclusion relation, defined in subsection 3.3, contains, for each cluster, the fuzzy set of clusters that contain it. We will use the notation $[I(S_i)](S_j) \doteq I(S_j, S_i)$ for the fuzzy set of clusters that contain cluster S_i, i.e. for the respective row of I, and the notation $I(s) \doteq I(\{s\})$ for the set of clusters that contain term s. We will use the term *context* of S_j and s for $I(S_j)$ and $I(s)$, respectively. Therefore, the context of a term contains the clusters whose terms are contained in the same documents with this term. Since we consider the terms of a cluster as having a semantic correlation, the clusters which form the context are the groups of semantically correlated terms which tend to be found in the same document with a given term, or set of terms. Thus, the context of the query can be used for query expansion.

Obviously, $S_1 \subseteq S_2 \implies I(S_1) \supseteq I(S_2)$, i.e. the presence of more terms will make the context narrower. Moreover, if $S_i = \{s_{ij}, j \in \mathbb{N}_{n_i}\}$, then $I(S_i) = \bigcap\limits_{j \in \mathbb{N}_{n_i}} I(s_{ij})$, i.e. the overall context is the intersection of the individual contexts.

Let us now extend the above definition of context to the case of fuzzy queries. Similarly to the above definitions, we define the context $K(q)$ of the query q as a fuzzy set on the set of clusters $\mathcal{P}(S)$, i.e. $K(q) = \sum\limits_i S_i / K(q)_i$.

First, we show that a direct extension of the above definition in the fuzzy case, for example $K^*(q) = \bigcap\limits_i q_i K(s_i)$, is not meaningful [16]. A low degree of importance q_i for the term s_i implies that the meaning of s_i is relatively insignificant for the query. On the other hand, it is implied by the above definition of K^* that a low value of q_i will narrow the context more than a high one; this is the opposite effect than what is desired.

In the following, we define the context $K(q)$ of the query q as the fuzzy intersection of the individual weighted contexts of each term :

$$K(q) = \bigcap_i K_q(s_i)$$

In order to achieve the desired effect, i.e. having a query term with low weight narrowing the context less than one with a hight weight, the following conditions must be satisfied, for the weighted context $K_q(s_i) = \sum_j S_j / K_q(s_i)_j$ of term s_i:

- if $q_i = 0$, then $K_q(s_i) = K(q \cap (S \backslash \{s_i\}))$ (no narrowing of context)
- if $q_i = 1$, then $K_q(s_i) = I(s_i) \cap K(q \cap (S \backslash \{s_i\}))$
- $K_q(s_i)_j$ decreases monotonically with q_i

Our approach is linear:

$$K_q(s_i)_j = 1 - q_i(1 - I(s_i)_j)$$

We will use this definition of context in subsection 4.3 in order to perform a context - aware query expansion. When the context contains high degrees for the clusters, the context will be important for query expansion. As a measure of this importance, we will use the height of the context $l = h(K(q))$; we will use the term *intensity* for l.

4.2 Handling of Terms in Query Expansion

In section 1, we explain that the search engine uses the query q and the document index D, which is a fuzzy relation between the set of terms S and the set of documents T, to produce the result r; r is a fuzzy set on T. When the query is comprised of a single term s, then the result is simply the respective row of D, i.e. $r(q) = D(s)$. When the query contains more than one terms, then the result is the set of documents that contain all terms, i.e. $r(q) = D(s_1) \cap D(s_2) \cap D(s_3)$.

In query expansion, we replace each term s with a set of terms $X(s)$; we will refer to this set as the *expanded term*. During evaluation of the query, we treat $X(s)$ considering a union operation, i.e. documents that match any term contained in $X(s)$ are selected. Therefore, in order to preserve the intersection operation among the original query terms, we need to expand each term separately.

4.3 Term Expansion

Using the above principle, we define, in this subsection, the expansion $X(s_i)$ of a single query term s_i as a fuzzy set on s, i.e. $X(s_i) = \sum_j s_j / x_{ij}$. We define it as a function of the query q, the context $K(q)$ of the query, and the set of valid clusters. The weight x_{ij} denotes the degree of significance of the term s_j in $X(s_i)$.

In a context – insensitive query expansion, i.e. when the intensity l of the query context is zero, the weight q_{ijk} must increase monotonically with respect to the following quantities, with respect to a cluster S_k:

- the weight q_i of term s_i in the query.
- the weight w_{ki} of term s_i in S_k
- the weight w_{kj} of term s_j in S_k

Based on the above, we define:

$$q_{ijk} \doteq q_i \cdot \min(w_{ki}, w_{kj})$$

For each cluster Sk, q_{ijk} denotes the minimum degree, to which s_j must participate in $X(s_i)$. Therefore, the degree with respect to all clusters, in a context – insensitive expansion, is:

$$q_{ij} = \max_{k \in \mathbb{N}_m} \{q_{ijk}\}$$

In a context – sensitive query expansion, the weight x_{ijk} increases monotonically, with respect to the degree to which the context of S_k is related to the context of the query. We will use the quantity

$$l_k \doteq \frac{h(K(q) \cap I(S_k))}{l}$$

as a measure of this relevance.

The following conditions must be satisfied by x_{ijk}:

- x_{ijk} increases monotonically with respect to q_{ijk}
- $l_k \to 1 \implies x_{ijk} \to q_{ijk}$
- $l = 0 \implies x_{ijk} \to q_{ijk}$
- $l \to 1 \implies x_{ijk} \to q_{ijk} l_k$

Following a linear approach, we obtain:

$$x_{ijk} \doteq (1 - l^{\alpha}(1 - l_k)) q_{ijk}$$

α is a positive parameter which controls the importance of context in query expansion. As α approaches zero, context is considered to the greatest extent. As α approaches infinity, the query expansion process becomes insensitive to context.

We can observe that the expanded query is a superset of the original one, regardless of the context intensity.

Table 1. The index. Rows correspond to terms and columns to documents. Blanks correspond to zeros.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	1	1	1	1	1	1	1	1	1	1	1	1	1				1	1	1
B				1	1	1					1	1							
C	1	1	1				1	1					1						
D									1	1							1	1	1
E				1	1	1						1							
F							1												
G								1											
H	1	1											1						
I	1	1	1	1	1	1								1	1	1			
J	1	1	1												1	1			
K	1			1	1	1								1					
L	1	1	1												1	1			

5 Simulation Example

In order to simulate the proposed methods, we provide an example of an index in Table 1. The documents of the index regard engines and airplanes. The index terms are: A="engine", B="rocket", C="internal combustion", D="external combustion", E="turbine", F="two–stroke", G="four–stroke", H="Diesel", I= "airplane", J="propeller", K="jet", L="propeller airplane".

The clusters of terms that our algorithm detects, excluding the singletons, are: AB,AC, AD, AI, AK, BE, CF, CG, CH, HK, IK, JL, ABE, ABK, ACH, ACI, AIK, BEI, BEK, BIK, CHK, CJL, HJL, HKL, IJL,ABEI, ABIK, AJKL, CHJL, CIJL, HIJL, IJKL, ACIJL, CHIJL, HIJKL, ACHIJL. The thresholds were $c_v = 0.1$ and $c_m = 0.9$.

Let the query be $q = A/1 + I/1$. The context–insensitive expansion of q is: $x_A = A/1 + B/0.25 + C/0.38 + D/0.31 + E/0.25 + H/0.19 + I/0.38 + J/0.16 + K/0.24 + L/0.16$ and $x_I = A/0.38 + B/0.27 + C/0.25 + E/0.27 + H/0.20 + I/1 + J/0.56 + K/0.56 + L/0.56$

The intensity of the query context is $l = 0.38$. The context–sensitive expansion of q, with $\alpha = \frac{1}{3}$ is: $x_A = A/1 + B/0.20 + C/0.27 + D/0.10 + E/0.16 + H/0.10 + I/0.38 + J/0.12 + K/0.19 + L/0.12$ and $x_I = A/0.38 + B/0.22 + C/0.16 + E/0.22 + H/0.12 + I/1 + J/0.36 + K/0.43 + L/0.36$.

It can be observed that term D="external combustion", which is related to engines but not in the context of airplane engines, is diminished in the context–sensitive expansion. This is derived from the index, which does not contain documents with both airplanes and external combustion engines.

6 Conclusions and Future Work

In this work, we consider term co-occurrence in documents in order to form groups of correlated terms. This clustering does not need as input the number of the clusters, neither does it produce a crisp hierarchy. We express the context of the query in terms of clusters and use it to expand the query in a context–sensitive manner. This statistical approach is useful when no knowledge about the terms is available.

In the proposed method, mainly linear approaches are applied, for the sake of simplicity. We believe that more general, non – linear approaches might be interesting to investigate. Moreover, the non-exponential computational complexity of the clustering algorithm must be confirmed. Finally, the result of the statistic query expansion can also be combined with the result of the semantic query expansion in an efficient manner.

References

1. Kraft D.H., Bordogna G. and Passi G., Fuzzy Set Techniques in Information Retrieval, in James C. Berdek, Didier Dubois and Henri Prade (Eds.) *Fuzzy Sets in Approximate Reasoning and Information Systems*, (Boston: Kluwer Academic Publishers, 2000).
2. G. Salton, M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.
3. Efthimiadis, E. N. Query expansion. In M. E. Williams (Ed.), *Annual review of information science and technology*, (vol. 31, pp. 121-187). Medford NJ: Information Today Inc. 1996
4. Harman, D. K. (1992). Relevance feedback revisited. In N. J. Belkin, P. Ingwersen and A. Mark Pejtersen (Eds.), SIGIR 92, *Proceedings of the 15th annual international ACM SIGIR Conference on research and development in information retrieval* (pp. 1-10). New York: ACM.
5. Salton, G. and Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41, 288-297.
6. W. Bruce Croft, Jinxi Xu, Corpus-specific stemming using word form co-occurence, in: *Proceedings of the Fourth Annual Symposium*, 1994.
7. N. J. Belkin, C. Cool, D. Kelly, S. -J. Lin, S. Y. Park, J. Perez-Carballo and C. Sikora, Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval, *Information Processing & Management*, Volume 37, Issue 3, May 2001, Pages 403-434
8. Miyamoto S., *Fuzzy sets in information retrieval and cluster analysis*, (Dordrecht/Boston/London: Kluwer Academic publishers, 1990)
9. Wen-Syan Li and Divyakant Agrawal, Supporting web query expansion efficiently using multi-granularity indexing and query processing, *Data & Knowledge Engineering*, Volume 35, Issue 3, December 2000, Pages 239-257
10. Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In H.-P. Frei, D. Harman, P. Schauble, & R. Wilkinson (Eds.), SIGIR '96, *Proceedings of the 19th annual international ACM SIGIR Conference on research and development in information retrieval* (pp. 4-11). New York: ACM
11. E.M. Voorhees, Query Expansion Using Lexical-Semantic Relations, in: *Proceedings of the 17th Annual International ACM SIGIR Conference*, Dublin, Ireland, 1994.
12. Kraft D.H., Petry F.E., Fuzzy information systems: managing uncertainty in databases and information retrieval systems, *Fuzzy Sets and Systems*, 90 (1997) 183-191, Elsevier.
13. Akrivas G., Stamou G., Fuzzy Semantic Association of Audiovisual Document Descriptions, *Proc. of Int. Workshop on Very Low Bitrate Video Coding (VLBV)*, Athens, Greece, Oct. 2001

14. Akrivas G., Stamou G. and Kollias S., Semantic Association of Multimedia Document Descriptions through Fuzzy Relational Algebra and Fuzzy Reasoning (submitted)
15. Klir G. and Bo Yuan, *Fuzzy Sets and Fuzzy Logic, Theory and Applications*, New Jersey, Prentice Hall, 1995
16. Akrivas G., Wallace M., Stamou G. and Kollias S., Context – Sensitive Semantic Query Expansion, *IEEE International Conference on Artificial Intelligence Systems AIS-02* (to appear)
17. ISO/IEC JTC 1/SC 29 M4242, *Text of 15938-5 FDIS Information Technology – Multimedia Content Description Interface – Part 5 Multimedia Description Schemes*, October 2001.

Intelligent Information Integration as a Constraint Handling Problem

Liviu Badea and Doina Tilivea

AI Lab, National Institute for Research and Development in Informatics
8-10 Averescu Blvd., Bucharest, Romania
badea@ici.ro

Abstract. Intelligent Information Integration (I^3) aims at combining heterogeneous and distributed information sources by explicitly representing and reasoning about their content, giving the user the illusion of interacting with a uniform system. In this paper we show how query planning in such a system can be reduced to a constraint handling problem. Conceptually, our approach relies on a generalized abductive reasoning mechanism involving so called partially open predicates, which support a seamless combination of backward (goal-directed) and forward reasoning. The original aspects of our approach consist in the early detection of (plan) inconsistencies using forward propagation of constraints as well as in the seamless interleaving of query planning and execution. Unlike other specialized query planning algorithms, for which domain reasoning and query planning are only loosely coupled, our encoding of source and domain models into Constraint Handling Rules allows us to fully and efficiently exploit existing domain knowledge. The ability to automatically derive source interactions from domain models (ontologies) enhances the flexibility of modeling.

1 Introduction and Motivation

The integration of hybrid modules, components and software systems, possibly developed by different software providers, is a notoriously difficult task, involving various complex technical issues (such as distribution, different programming languages, environments and even operating systems) as well as conceptual problems (such as different data models, semantic mismatches, etc.).

Solving the conceptual problems requires the development of an explicit (declarative) common knowledge model of the systems to be integrated. Such a model is used (by a so-called *mediator* [Wie92]) not only during the development of the integrated system, but also at query time, when it can be manipulated by a *query planner* to solve problems that could not have been solved by any of the specific information sources alone and might even not have been foreseeable by the system integrator.

The *query planner* of the mediator transforms a user query into a sequence of information-source specific queries that jointly solve the original query. The *query execution* module performs the actual source accesses, while the *result integration* component combines the results obtained from the information sources in a globally consistent result.

In this paper we present an original encoding of *query planning* as a *constraint handling problem*. Conceptually, our approach relies on a generalized abductive reasoning mechanism involving so called *partially open predicates*, which support a seamless combination of backward (goal-directed) and forward reasoning. The original aspect of our approach consists in the early detection of (plan) inconsistencies using forward propagation of constraints. This is especially important since the main purpose of query planning is to prevent as many potential failures as possible at the (early) planning stage, i.e. before actually accessing the information sources. The query planner has been implemented using Constraint Handling Rules (CHR) [Fr98] and used within a complete Information Integration System, called SILK [S].

Unlike other specialized query planning algorithms, for which domain reasoning and query planning are only loosely coupled, our encoding of source and domain models into Constraint Handling Rules allows us to fully and efficiently exploit existing domain knowledge. [1]

CHRs also enable a natural interleaving of *query planning* with *execution*. It would be unrealistic to assume that the query planning stage will be able to construct foolproof plans, i.e. plans that do not fail (especially because of the incomplete knowledge available to the planner). Therefore, we should expect failures and hence provide mechanisms for *replanning* during execution.

As opposed to many other more procedural approaches to information integration (e.g. [GPQ97]), we present an approach to I^3 using logic not just for modeling information sources and domains, but also for *reasoning* about them, the main advantage of such an approach being its flexibility. (For efficiency reasons, we use a Constraint Logic Programming environment.)

2 Modeling Information Sources

While the data content of an information source can be represented by a number of source predicates *s*, the user might prefer to have a *uniform* interface to *all* the sources instead of directly querying the sources *s*. In fact, she may not even *want* to be aware of the structure of the information sources. Therefore, the mediator uses a uniform knowledge representation language in terms of so-called *"base predicates"*, which represent the user's perspective on the given domain.

There are two main viewpoints on representing sources and their interactions: *"Global as View"* and *"Local as View"* [Lev00]. In the *Global As View* (*GAV*) approach, interactions between sources have to be described explicitly for every combination of sources that interact. For example

$$s_1(Empl,Department,\ldots) \wedge s_2(Empl,Project,\ldots) \rightarrow proj\_dept(Project, Department). \qquad (1)$$

[1] For example, the Information Manifold [LRO96] uses a specialised query planning algorithm only loosely coupled with the domain reasoning module (CARIN). A tighter integration of domain reasoning with query planning allows us to discover inconsistent plans earlier.

expresses the fact that a particular combination of s_1 and s_2 has property *proj_dept*(*Project, Department*) (which is a "base predicate").

The need to consider all relevant combinations of sources has the obvious drawback that adding a new source is complicated, since it involves considering all potential interactions with the other sources. (There can be an exponential number of such interactions.)

On the other hand, in the *Local As View* (*LAV*) approach, the interactions between sources are being handled by the query planner at the mediator level. All that needs to be done when adding a new source is to describe all its relevant properties (in the mediator language, i.e. in terms of base predicates). For example, one would independently describe the sources s_1 and s_2 using the implications

$s_1(Employee, Department, \ldots) \rightarrow emp\_dept(Employee, Department)$

$s_2(Employee, Project, \ldots) \rightarrow emp\_proj(Employee, Project)$

and delegate the detection of potential interactions between them to the query planner, which would need domain specific descriptions of the base predicates, such as

$$emp\_dept(Emp,Dept) \wedge emp\_proj(Emp,Proj) \rightarrow proj\_dept(Proj, Dept). \quad (2)$$

Note that the LAV approach only shifts the description of source interactions from the source level (1) to the level of base predicates (2). This is not just a simple technical trick, since descriptions of the form (2) would be part of the *domain ontology* (which would be developed once and for all and therefore wouldn't change as new sources are added). In the LAV approach, the description of the sources is easier, since the knowledge engineer only has to describe the properties of the newly added source in terms of base predicates, without having to worry about potential source interactions. This latter task is delegated to the query planner.

The query planner presented in this paper can deal with both GAV and LAV approaches. However, we encourage the use of LAV, since it spares the knowledge engineer the effort of explicitly determining and representing the source interactions.

We start by giving the intuition behind our modeling of source descriptions and their encoding using Constraint Handling Rules (CHR) and Constraint Logic Programming (CLP).

Constraint Handling Rules (see [Fr98] for more details) represent a flexible approach to developing user-defined constraint solvers in a declarative language. As opposed to typical constraint solvers, which are black boxes, CHRs represent a 'no-box' approach to CLP.

CHRs can be either *simplification* or *propagation* rules.

A *simplification* rule *Head* <=> *Guard* | *Body* replaces the head constraints by the body provided the guard is true (the *Head* may contain multiple CHR constraint atoms).

Propagation rules Head ==> *Guard* | *Body* add the body constraints to the constraint store without deleting the head constraints (whenever the guard is true). A third, hybrid type of rules, *simpagation rules* $Head_1 \backslash Head_2$ <=> *Guard* | *Body* replace $Head_2$ by *Body* (while preserving $Head_1$) if *Guard* is true. (Guards are optional in all types of rules.)

Now, a Local As View source description like $s(X) \rightarrow b_1(X), b_2(X), \dots$ (3)
is implemented using a CHR *propagation rule* $s(X) ==> b_1(X), b_2(X), \dots$ (4)
which ensures that every time a query is reduced to a source predicate $s(X)$, the source's properties are automatically propagated (thereby enabling the discovery of potential inconsistencies). Propagation rules however, capture only one direction of the implication (3). Since they are unable to deal with the backward direction of (3), they are inappropriate for regressing a goal in terms of base predicates to a subgoal in terms of the source predicate. This has to be explicitly implemented using *goal regression rules* of the form $b_i(X) \textbf{:-}\ s(X).$ (5)

Unfortunately, this simple approach may loop, since it doesn't distinguish between *goals* (which have to be achieved) and *facts* that are simply propagated forward. We address this problem by separating *goals* (denoted simply as ***Gp***) from *facts*, which are asserted to hold, represented as ***Hp***. (In the implementation, ***Hp*** is represented as a CHR constraint, *holds*(*p*), while goals ***Gp*** are represented explicitly as *goal*(*p*).)
Propagation rules involve *facts*: $\boldsymbol{H}s(X) ==> \boldsymbol{H}b_1(X), \boldsymbol{H}b_2(X), \dots$ (6)
while goal regression rules involve *goals*: $\boldsymbol{G}b_i(X) <=> \boldsymbol{G}s(X).$ (7)

The following schema (w.r.t. *p*) takes care of the consistency between goals and facts that simply hold (since goals will be achieved eventually, they also have to hold):

$$\boldsymbol{G}p ==> \boldsymbol{H}p \qquad (8)$$

Note that rule (8) should have a higher priority than (7) (***Hp*** should be propagated before the goal is replaced by a subgoal in (7)).

We formalize the above distinction between goals and facts by introducing the notion of *open predicates*. The query planner essentially interleaves two reasoning phases: *goal regression* and *forward propagation* of properties ("saturation"). While goal regression uses the current "closure" (body) of a given base predicate, forward propagation may produce *new instances* of the base predicate (which have to hold for the partial query plan to be consistent). Thus, we need to be able not only to refer to the closure of a given predicate, but also to allow it to be "open".

2.1 Open Predicates

In Logic Programming, normal predicates are *closed*: their definition is assumed to be complete (Clark completion). On the other hand, abducibles in Abductive Logic Programming (ALP) [KKT98] are *completely open*, i.e. they can have any extension consistent with the integrity constraints. For dealing in a formal manner with forward propagation rules in an abductive framework, we need to allow a generalization of abducibles, namely (partially) *open predicates*.

Unlike normal abducibles, open predicates can have definitions $p \leftarrow Body$, but these are not considered to be complete, since during the problem solving process we can add (by forward propagation) new abductive instances of *p*. The definition of an open predicate therefore only partially constrains its extension.

In our CHR embedding of the abductive procedure we use two types of constraints, ***Gp*** and ***Hp***, for each open predicate *p*. While ***Hp*** represents facts explicitly propagated (abduced), ***Gp*** refers to the *current closure* of the predicate *p* (i.e. the explicit definition of *p together* with the explicitly abduced literals ***Hp***). Thus, informally[2], the extension of *p* is ***Gp*** $= def(p) \vee$ ***Hp***.

While propagating ***Hp*** amounts to simply assuming *p* to hold (abduction), propagating ***Gp*** amounts to trying to prove *p* either by using its definition *def*(*p*), or by reusing an already abduced fact ***Hp***. This distinction ensures that our CHR embedding conforms to the usual '*propertyhood view*' on integrity constraints. In fact, our description rules presented below can be viewed as a generalization of abductive logic programs with integrity constraints interpreted w.r.t. the 'propertyhood view'[3].

Definition 1. $M(\Delta)$ is a *generalized stable model* of the abductive logic program $\langle P,A,I \rangle$ for the abductive explanation $\Delta \subseteq A$ iff (1) $M(\Delta)$ is a stable model of $P \cup \Delta$, and (2) $M(\Delta) \models I$.

The distinction between propagating ***Hp*** and ***Gp*** respectively can be explained best using an example. Earning a certain income should propagate the *obligation* of having one's taxes paid (represented by the closure ***G****taxes_paid*). On the other hand, one could imagine a scenario in which the taxes are paid ***H****taxes_paid*, without having the corresponding obligation (goal) ***G****taxes_paid* (for example, as a side-effect of a different goal).

The use of *open predicates* allows mixing *forward propagation* of abduced predicates ***Hp*** with *backward reasoning* using the closures ***Gp***. Forward propagation can be implemented using CHR propagation rules, while backward reasoning involves unfolding predicates with their definitions. The definition *def*(*p,Body*) of a predicate *p* is obtained by Clark completion of its 'if' definitions. For each such predicate we will have an *unfolding rule* (similar to *p* **:-** *Body*) implemented as a CHR simplification rule: ***Gp*** <=> *def*(*p,Body*) | ***G****Body*,

but also a CHR simpagation rule[4] for matching a goal ***Gp*** with any existing abduced facts ***Hp***: $\boldsymbol{H}p(X_1) \setminus \boldsymbol{G}p(X_2) \Longleftrightarrow X_1{=}X_2\ ;\ X_1{\neq}X_2,\ \boldsymbol{G}p(X_2).$ (re)

This rule should have a higher priority than the unfolding rule in order to avoid reachieving an already achieved goal. Note that, for completeness, we are leaving open the possibility of achieving ***Gp***(X_2) using its definition or reusing other abduced facts.

Our treatment of open predicates ***Gp*** $= def(p) \vee$ ***Hp*** is subtly different from the usual method [KKT98] of dealing with (partially) open predicates *p*, where a new predicate name *p'* (similar to our ***Hp***) is introduced and the clause $p \leftarrow p'$ is added to the definition of *p*: $\{p \leftarrow Def,\ p \leftarrow p'\}$. (*)

The difference is that whenever we refer to ***Gp*** we are implicitly trying to prove *p*, either by using its definition *def*(*p*) or by reusing an already abduced fact ***Hp***, *but without allowing such an* ***Hp*** *to be abduced in order to prove* ***Gp*** (whereas in (*)

[2] For lack of space, we defer the formalization of these notions to the full paper.

[3] The detailed presentation of the semantics is outside the scope of this paper and will be pursued elsewhere. Briefly, the goal regression rules play make up the program *P*, the sources are regarded as (temporary) abducibles *A*, while the forward propagation rules play the role of the ALP integrity constraints *I*. Our notion of integrity constraints is however more general than that used in ALP.

[4] The rule is more complicated in practice, due to implementation details.

treating $p \leftarrow p'$ as a *program clause* would allow p' to be abduced when trying to prove p). This is crucial for ensuring a correct distinction[5] between goals ***G****p* and abducibles ***H****p* mentioned above (otherwise we would treat the propagation of goals and obligations incorrectly). Without making this distinction, *we wouldn't even be able to refer to the current closure of p.*

Besides source descriptions (3), we also have to be able to represent logical constraints between base predicates (for example concept hierarchies, integrity constraints, etc.). The same implementation strategy can be used in this case. For example, an implication of the form $b_1(X,Y) \wedge b_2(Y,Z) \rightarrow b(X,Z)$ can be represented using a CHR propagation rule $\boldsymbol{H}b_1(X,Y)$, $\boldsymbol{H}b_2(Y,Z)$ ==> $\boldsymbol{H}b(X,Z)$ and a goal regression rule $\boldsymbol{G}b(X,Z)$ **<=>** $\boldsymbol{G}b_1(X,Y)$, $\boldsymbol{G}b_2(Y,Z)$.

Example 1. Assume that in a software company we want to allocate qualified personnel to a new project while not exceeding a given budget. Assume that the company's information system contains separate databases for each department (in our case ***administrative*** and ***development***, which are regarded as source predicates), as well as a database of ***student***s (part-time employees) who could be employed in the project, in case the overall project budget is exceeded by using full-time personnel (students are assumed to have a lower salary).

The end user of the integrated system (for example the company's manager) would use ***base predicates*** such as: *employee*(*Employee*), *qualification*(*Employee, Qualification*), *salary*(*Employee, Salary*). The following description rules are used to represent ***domain-specific knowledge*** (descriptions for base predicates):

qualification(*Empl*, *'C'*) → *programmer*(*Empl*) (e1)
qualification(*Empl*, *'Prolog'*) → *programmer*(*Empl*) (e2)
programmer(*Empl1*), *non_programmer*(*Empl2*) → *Empl1* ≠ *Empl2* (e3)

The following *description rules* are used to characterize the ***source predicates***:

administrative(*Empl*, *Qual*, *Sal*) → *employee*(*Empl*), *qualification*(*Empl*, *Qual*), *salary*(*Empl*, *Sal*), *non_programmer*(*Empl*). (e4)

development(*Empl*, *Qual*, *Sal*) → *employee*(*Empl*), *qualification*(*Empl*, *Qual*), *salary*(*Empl*, *Sal*), *Sal* > *2000*, *programmer*(*Empl*). (e5)

student(*Empl*, *Qual*) → *employee*(*Empl*), *qualification*(*Empl*, *Qual*), *salary*(*Empl*, *Sal*), *Sal* = *1000*, *programmer*(*Empl*). (e6)

The above description rules assert that:

- people with qualifications in *C* and *Prolog* are programmers, while programmers and non-programmers are disjoint concepts.
- administrative employees are non-programmers
- employees from the development department are programmers (with a salary over 2000)
- students are also programmers and are assumed to have a fixed salary of 1000.

[5] This distinction is essential only for *partially* open predicates and not for completely open predicates (abducibles).

The user might want to allocate two employees *Empl1* and *Empl2* with qualifications *'C'* and *'Prolog'* such that the sum of their salaries does not exceed 3500 (see continuation of the Example in Section 3.3).

3 Query Planning with Constraint Handling Rules

In the following, we concentrate on a more detailed description of the representation language for source descriptions and domain models, as well as on their implementation using Constraint Handling Rules.

3.1 Base Predicates, Source Predicates, and Description Rules

The content of the various information sources is represented by so-called *source predicates*, which will be described at a declarative level in terms of the so-called *"base predicates"*. More precisely, we distinguish between *content* predicates and *constraint* predicates.

Content predicates (denoted in the following with *p*, *q*, possibly with subscripts) are predicates which directly or indirectly represent the *content* of information sources. They can be either *source* predicates or *base* predicates.

Source predicates (denoted with *s*) directly represent the content of (part of) an information source (for example, a table in a relational database, or the services provided by the interface of a procedural application). Their definitions (bodies) are not explicit at the mediator level, but can be accessed by querying their associated information source.

Base predicates (denoted with *b*) are user-defined predicates for describing the domain, as well as the information content of the sources.

As opposed to content predicates, *constraint predicates* (denoted by *c*) are used to express specific constraints on the content predicate descriptions.

For example, a source *s* containing information about underpaid employees (with a salary below 1000), would be described as:

$s(Name, Salary) \rightarrow employee(Name) \wedge salary(Name, Salary) \wedge Salary<1000.$

(In the above, *s* is a source predicate, *employee* and *salary* are base predicates, while '<' is a constraint predicate.)

Constraint predicates can be either *internal* (treatable internally by the query engine of the source), or *external* (constraints that can only be verified at the mediator level, for example by the built-in constraint solvers of the host CLP environment). Constraints treatable *internally* by the sources can be *verified* at the source level (by the query engines of the sources), but they are also *propagated* at the mediator (CLP) level. Constraints treatable only *externally* need to be both verified and propagated at the mediator level.

A *complete* description of the source predicates in terms of base predicates is neither possible nor useful, since in general there are too many details of the functioning of the sources that are either unknown to the user, or irrelevant from the point of view of the application. Thus, instead of *complete* (iff) descriptions, we shall specify only approximate (necessary) definitions of the source predicates in terms of base predicates (thus, only the relevant features of the sources will be encoded).

In the following, we use a *uniform* notation for the domain and source **description rules**:

$$\forall \overline{X}.\ p_1(\overline{X}_1) \wedge \ldots \quad \rightarrow \quad \exists \overline{Z}.\ b_1(\overline{Y}_1) \wedge \ldots \ \wedge s_i(\overline{Y}_i) \wedge \ldots \ \wedge c_j(\overline{Y}_j) \wedge \ldots \tag{9}$$

where $\overline{X} = \bigcup_i \overline{X}_i,\ \ \overline{Y} = \bigcup_j \overline{Y}_j,\ \ \overline{Z} = \overline{Y} - \overline{X}$ (with variable tuples viewed as sets).

Description rules are necessary definitions of (combinations of) source or base predicates in terms of base predicates and constraints. (*Source descriptions* are special cases of description rules. *Integrity constraints* are description rules with only constraints – typically *'fail'* – in the consequent.[6])

Normally, source descriptions are *coarser grained* than the actual information content of the sources (they abstract away the irrelevant details). However, there are situations in which a source description is *finer grained* than the explicit source content, typically due to (meta-level) knowledge about the content of the source that is not explicitly contained in the source.

Example 2. A source of cheap cars may not contain the actual prices of the cars, but we may possess the (meta-level) knowledge that these prices are below 10000:

$s(Model) \rightarrow \exists P.\ car(Model) \wedge price(Model, P) \wedge P<10000.$

Such conceptual descriptions that are finer grained than the source content involve existential variables in the consequent and are, in a sense to be made precise shortly, "nondecomposable".

Occurrences of source predicates in the consequent do not represent updates to the corresponding source. Rather, they are typically used for *conversions* from the format used in sources to the common format used at the mediator level, or more generally for expressing *general constraints* that are encoded in the given source predicate. These need to be *verified* rather than propagated.

Example 3. Prices of cars may be stored in a different currency, for which a source *s_exchange* (functioning as a conversion table) is available:

$s(Model, Price) \rightarrow \exists P.\ car(Model) \wedge s\_exchange(Price, P) \wedge price(Model, P).$

Description rules provide a convenient and very expressive way of *describing* sources as well as the domain knowledge.

---

[6] An integrity constraint of the form $\leftarrow p(X),q(X)$ could be written either as $p(X1),q(X2) \Longrightarrow X1 \neq X2$ or as $p(X),q(X) \Longrightarrow fail$.

3.2 Implementing Description Rules in CHR

Due to their flexibility and declarative nature, constraint Handling Rules (CHRs) [Fr98] represent an ideal framework for implementing the reasoning mechanism of the query planner.

A description rule of the form (9) will be encoded in CHR using

- *goal regression rules*: for reducing queries given in terms of base predicates to queries in terms of source predicates, and
- *propagation rules*: for completing (intermediate) descriptions in order to allow the discovery of potential inconsistencies.

An implication of the form $p \rightarrow q_1 \wedge q_2 \wedge \ldots$ is *decomposable* into a conjunction of implications
$(p \rightarrow q_1) \wedge (p \rightarrow q_2) \wedge \ldots$ whenever the consequent includes no existential variables. In the general case, however, we need to Skolemize the description rules (9) before decomposing them (decomposition being necessary for generating goal regression rules):

$$\forall \overline{X}.\ p_1(\overline{X}_1) \wedge \ldots \rightarrow \ldots \wedge sk(k, Z_k, \overline{X}) \wedge \ldots b_1(\overline{Y}_1) \wedge \ldots s_i(\overline{Y}_i) \wedge \ldots c_j(\overline{Y}_j) \wedge \ldots \quad (10)$$

where $sk(k, Z_k, \overline{X})$ denotes the fact that the existential variable Z_k depends on the variable tuple $\overline{X}$. Essentially, this is a way of writing the Skolem term $Z_k = f_k(\overline{X})$ using a predicate *sk* instead of a function f_k. This simplifies our dealing with such Skolems in our CHR implementation, in which *sk* is a CHR constraint subject to the following propagation rule, which ensures that $f_k(X1)=f_k(X2)$ if $X1=X2$:

$sk(K,Z1,X),\ sk(K,Z2,X)$ ==> $Z1=Z2$. (sk)

3.2.1. Goal Regression Rules

Now, (10) is decomposable into separate implications

$$\forall \overline{X}.\ p_1(\overline{X}_1) \wedge \ldots \quad \rightarrow \quad \ldots \wedge sk(k, Z_k, \overline{X}) \wedge \ldots \quad \wedge b_l(\overline{Y}_l). \quad (11)$$

for each base predicate b_l occurring in the consequent of (10) (in the consequent of (11) we only keep the Skolems for which $Z_k \in \overline{Y}_l$).

(11) can easily be viewed as a goal regression rule for b_l:

$$\boldsymbol{G}\, b_l(\overline{Y}_l) \quad \texttt{<=>} \quad \ldots,\ sk(k, Z_k, \overline{X}), \ldots,\ \boldsymbol{G}\, p_1(\overline{X}_1), \ldots \quad (12)$$

Since (10) is *not* a sufficient definition for the *source* predicates s_i appearing in its consequent (s_i playing here the role of constraints), we do not generate goal regression rules of the form (11) or (12) for s_i. Neither do we produce such rules for the constraints c_j.

3.2.2 Propagation Rules

A set of subgoals in terms of source predicates (induced by backward chaining from the user query using the goal regression rules) may not necessarily be consistent. Applying (10) as forward propagation rules ensures the completion ("saturation") of the (partial) query plan and enables detecting potential conflicts *before* actually accessing the sources.

As base predicates p are subject not just to normal backward reasoning, but also to forward propagation, they will be treated as open predicates. The forward propagation rules will thus involve their "open" part $\boldsymbol{Hp}$:

$$\boldsymbol{Hp}_1(\overline{X}_1) \wedge \ldots \texttt{==>} \ldots, sk(k, Z_k, \overline{X}), \ldots, \boldsymbol{Hb}_1(\overline{Y}_1), \ldots, \boldsymbol{Gs}_i(\overline{Y}_i), \ldots, \mathbf{C}c_j(\overline{Y}_j), \ldots \qquad (13)$$

Note that source predicate occurrences $s_i(\overline{Y}_i)$ in the consequent refer to the "closure" $\boldsymbol{Gs}_i$ of s_i (rather than their open parts $\boldsymbol{Hs}_i$), since sources in the consequent are used as constraints to be *verified*, rather than propagated.

We have already mentioned the fact that we have to distinguish between goals involving a given predicate p (which will be treated by a mechanism similar to the normal Prolog backward chaining mechanism) and the instances $\boldsymbol{Hp}$ of p, which trigger forward propagation rules. Operationally speaking, while goals $\boldsymbol{Gp}$ are "consumed" during goal regression, the fact that p holds should persist even after the goal has been achieved, to enable the activation of the forward propagation rules of p. Goals p will therefore have to propagate $\boldsymbol{Hp}$ (using rule (8)): $\boldsymbol{Gp}$ ==> $\boldsymbol{Hp}$ before applying the goal regression rules for p: $\boldsymbol{Gp}$ <=> $body(p)$.

A predicate p for which $\boldsymbol{Hp}$ is propagated *only* by rule (8) is *closed* ($\boldsymbol{Gp} = body(p)$) since all propagated $\boldsymbol{Hp}$ instances will verify $body(p)$. Propagating $\boldsymbol{Hp}$ instances in the consequents of other propagation rules makes p an open predicate.

$\mathbf{C}c_j$ in (13) represent constraint predicate calls (since our system is implemented in the CLP environment of Sicstus Prolog [Sics], we assume the availability of host constraints solvers for the usual constraint predicates).

Source and domain models are described using rules of the form (9), which are then automatically translated by the system into CHR goal regression and propagation rules (12) and (13). (The additional problem-independent rules (8), (re) and (sk) are used for obtaining the complete CHR encoding of a model.)

3.3 Source Capabilities and Query Splitting

Instead of directly executing source predicate calls (by actually accessing the sources), sub-goals $s_i(\overline{Y}_i)$ are delayed to enable their aggregation into more specific sub-queries, thereby transporting less tuples at the mediator level. The goal regression rules for source predicates s post constraints of the form $\mathbf{S}s$, which denote the delayed source call: $\boldsymbol{Gs}(Y)$ <=> $\mathbf{S}s(Y)$.

A *query plan* consists of a number of such source predicate calls $\mathbf{S}s_i(\overline{Y}_i)$ as well as additional constraints $\mathbf{C}c_j(\overline{Y}_j)$ (propagated by forward rules (13)). Note that both types of constraints, $\boldsymbol{Hs}$ and $\mathbf{S}s$, are needed, the latter being "consumed" (deleted) after query execution (so that the source will not be queried again with *s(Y)*). On the other hand, ***Hs*** should persist even after executing the source access, because of potential interactions with constraints that may be propagated later on.

Information sources are viewed as *collections of source predicates* that can be accessed via a specialized query interface. (Such information sources can be databases, functional applications, etc.) The query planner reduces a query formulated in terms of base predicates to a query in terms of source predicates and constraints. However, since such a "global" query can involve source predicates from *several* information sources, it will have to be to *split* into sub-queries that can be treated by the separate information sources. Since each information source may have its own query processor, we need to explicitly represent the *capabilities* of these query processors. For example, a specific database interface may not be able to deal with arbitrary joins, may allow only certain types of selections and may also require certain parameters to be inputs (i.e. known at query time). Dataflow issues such as input-output parameters are especially important in the case of procedural applications.

Our *query splitting* algorithm (also implemented in CHR) incrementally selects the source predicate constraints ***Ss*** (from the global query plan) that can be dealt with by a given source. In doing this, it separates the constraints that can be treated *internally* by the source's query processor from the ones that need to be treated *externally* (at the level of the mediator). In order to minimize the answer sets of queries (which need to be transferred to the mediator level), the source queries are made as *specific* as possible. This is achieved by delegating to the source's query processor as many constraints as possible. (Only the ones that are not treatable internally, are dealt with externally.)

Example 1 (continued) In the following, we present the CHR encoding of Example 1 above, as well as the functioning of the latter as an abductive constraint-based query planner. (We recall that, for simplicity, we have used in the above ***Hp*** as a shorthand notation for the CHR constraint *holds(p)* and ***Gp*** as an abbreviation for the CHR constraint *goal(p)*. ***Cp*** represent constraint predicates calls.)

H*student*(*Empl*, *Qual*) ==> ***H****employee*(*Empl*), ***H****qualification*(*Empl*, *Qual*), (e6-p)
    sk(*1,Sal,*[*Empl,Qual*]), ***H****salary*(*Empl*, *Sal*), ***C***(*Sal = 1000*), ***H****programmer*(*Empl*).

G*employee*(*Empl*) <=> ***G****student*(*Empl*, *Qual*). (e6-g1)
G*qualification*(*Empl*, *Qual*) <=> ***G****student*(*Empl*, *Qual*). (e6-g2)
G*salary*(*Empl*, *Sal*) <=> *sk*(*1,Sal,*[*Empl,Qual*]), ***G****student*(*Empl*, *Qual*). (e6-g3)
G*programmer*(*Empl*) <=> ***G****student*(*Empl*, *Qual*). (e6-g4)

(e6-p) is the forward propagation rule corresponding to the description rule (e6), while (e6-g1)–(e6-g6) are its associated goal regression rules. (The CHR encodings for the rest of the descriptions rules (e1) – (e5) are very similar and are skipped for brevity.) Besides the above problem-specific rules, we also use the general rules (8), (re) and (sk) (in this order, and taking precedence over the problem-specific rules).

The query from Example 1

query(*Empl1,Empl2*) **:-** ***G***((*employee*(*Empl1*), *qualification*(*Empl1,'C'*), *salary*(*Empl1,Sal1*),
    employee(Empl2), qualification(Empl2,'Prolog'), salary(Empl2,Sal2), Sal1+ Sal2<3500)).

will first trigger the goal reduction rule for *employee*, regressing the goal ***G****employee*(*Empl1*) to ***G administrative***(*Empl1, 'C', Sal1*). The latter would then propagate with (e4) ***H****non_programmer*(*Empl1*)[7] (among others). The next goal, ***G****qualification*(*Empl1, 'C'*), will then propagate with (e1) ***H****programmer*(*Empl1*), which will produce an inconsistency with the previously propagated ***H****non_programmer*(*Empl1*) (using (e3)). This will trigger backtracking to the goal ***G****employee*(*Empl1*), which will now be reduced to the source predicate ***G development***(*Empl1, 'C', Sal1*) (propagating ***H****programmer*(*Empl1*), that is consistent with the next goal ***G****qualification*(*Empl1, 'C'*)). ***H development***(*Empl1, 'C', Sal1*) also propagates (with (e5)) ***C***(*Sal1 > 2000*).

A similar chain of reasoning steps will be performed for the sub-goals involving *Empl2*, leading to the source access ***development***(*Empl2, 'Prolog', Sal2*) and the constraint *Sal2 > 2000*. But unfortunately, the constraint *Sal1+Sal2<3500* is now violated by *Sal1>2000, Sal2>2000*, leading to a backtracking step in which the subgoal ***G****employee*(*Empl2*) is solved by accessing the source ***student***(*Empl2, 'Prolog'*), whose salary *Sal2=1000* is consistent with the constraint *Sal1+Sal2<3500*. All the above reasoning steps belong to the query *planning* phase. The resulting plan (consisting of source accesses and constraints)

development(Empl1,'C',Sal1), student(Empl2,'Prolog'), Sal2=1000, Sal1+Sal2<3500

will be split into queries for the separate sources (***development*** and ***student*** being assumed to be tables in different databases). While doing this, the constraints treatable internally by the various sources will be attached to the corresponding source queries. In the above example, the constraint *Sal1+ Sal2<3500* involves variables of two separate sources and will not be directly treatable by any of the sources. However, since *Sal2=1000*, it will be simplified to *Sal1<2500*, which is treatable internally by the first source. Therefore, the execution of the plan above will consist in querying the first source with ***development***(*Empl1, 'C', Sal1*), *Sal1<2500* and the second with ***student***(*Empl2, 'Prolog'*).

Note that querying the first source with the whole subplan ***development***(*Empl1, 'C', Sal1*), *Sal1<2500* (instead of eagerly querying it with ***development***(*Empl1, 'C', Sal1*) without bothering to construct plans), will typically transport much less tuples from the sources to the mediator (because *Sal1 < 2500* acts as a filter!).

Planning and execution will be *interleaved* in a seamless manner. For example, the second sub-query ***student***(*Empl2, 'Prolog'*) might fail (at query time we might discover that the database contains no students qualified in *Prolog*). One would then have to backtrack to the first subgoal ***G****employee*(*Empl1*) and obtain it from ***student***(*Empl1, 'C'*), presumably allowing ***G****employee*(*Empl2*) to be solved by using a more expensive permanent employee from ***development***(*Empl2, 'Prolog', Sal2*).

Note that if we knew (in the source description (e6)) that students do not know *'Prolog'*, we could have obtained the correct solution without accessing the source ***student***. This remark shows that source descriptions are very useful for pruning the search space at planning time (before actually querying the sources).

As far as we know, our sophisticated use of constraints for query planning and execution outperforms all existing query planning systems due to the *early detection of inconsistencies* as well as due to the *seamless interleaving of planning with execution*.

[7] To be more precise, ***G administrative***(*Empl1, 'C', Sal1*) propagates with (8) ***H******administrative***(*Empl1, 'C', Sal1*), which in turn propagates ***H****non_programmer*(*Empl1*) (using the forward propagation rule associated to (e4)).

3.4 Correctness and Completeness

Our query answering approach is weakly correct, complete and terminates for a set of acyclic description rules. To explain weak correctness, we need to consider the cases in which the sources may violate the integrity constraints. (This may be due to practical difficulties in maintaining the joint consistency of several distributed information sources, updated by different administrators.) In such a case, we do not simply report a global inconsistency and give up answering queries. Instead, we view integrity constraints as prescriptions on all possible query *answers*, rather than as constraints on the sources themselves (the latter viewpoint being difficult to enforce in practice).

In other words, we avoid testing the joint consistency of the sources (which is computationally very expensive anyway) and impose the integrity constraints only on the query answers.

Definition 2. An answer to a query is *weakly correct* iff the associated tuples retrieved from the sources verify the integrity constraints. (I.e. any potential violations of the integrity constraints involve source tuples that are not used in the query answer.)

Example 4. Assuming we are dealing with the following integrity constraints on the sources s1, s2 and s3:

Hs1(*X1*), ***Hs2***(*X2*) ==> *X1*=*X2*.
Hs1(*X1*), ***Hs3***(*X3*) ==> *X1*=*X3*.

the query ?- *s1*(*X1*), *s2*(*X2*) may return *X1*=a, *X2*=a, which is a weakly correct answer, since it verifies the first IC. But the answer *X1*=a, *X2*=b is incorrect, since it violates the first IC. Note that *X1*=a, *X2*=a is a weakly correct answer for the query above, even if we have a tuple *s3*(c) that potentially violates the second IC. (However, this inconsistency between *s1* and *s3* is irrelevant from the point of view of the given query and should not affect the answer.)

4 Concluding Remarks and Related Work

An exhaustive comparison with other information integration systems is impossible, due to lack of space. Briefly, while database oriented approaches to integration (such as *multi-databases* and *federated databases*) use *fixed* global schemas and are appropriate only if the information sources and users do not change frequently, we deal with *dynamically evolving* schemas (especially if the LAV modeling approach is employed). On the other hand, more *procedural* intelligent information integration approaches, like TSIMMIS [GPQ97] and even some declarative systems like MedLan [AART97] or HERMES [Sub], use explicit query reformulation rules[8], but *without the equivalent of our forward propagation rules* (which allow an early discovery and pruning of inconsistent plans before query execution). Our approach is closer to the more declarative systems like SIMS [AKH96], Information Manifold [LRO96] and Infomaster [DG97].

[8] Such query templates correspond to our goal regression rules.

However, while the Information Manifold (IM) uses a special purpose query planning algorithm, we are using a general Constraint Handling module, which allows an easy development of much more *flexible* intelligent information integration frameworks (the descriptions are declarative, while also allowing efficient reasoning about them). Also, the IM query planning algorithm cannot be easily extended to deal with Global as View descriptions (in which interactions between sources are explicit)[9]. Queries involving source descriptions with existential variables are also not treated completely (see the remark in Section 5 of [DG97b]). Additionally, our use of an optimized constraint propagation mechanism (like the one provided by CHR) may help *discover inconsistencies earlier* than in IM (which checks for consistency *all* possible combinations of sources from the buckets)[10].

On the other hand, IM allows using a description logic (DL) in source descriptions. Such DL descriptions can also be reformulated using our description rules. However, we cannot guarantee *complete* reasoning with such descriptions (on the other hand, *full* CARIN is highly intractable in the worst case). IM also doesn't allow reasoning with integrity constraints. In order to guarantee polynomial complexity of the algorithms for dealing with source capabilities, IM uses a more limited source capabilities language than ours (however, the cases of intractability may not appear in real-world cases).

Our approach is also similar to Infomaster (although few details are available in the published papers [DG97,DG97b]), the main difference being that we are using CHRs for reasoning, while Infomaster uses a model elimination theorem prover. (Although Infomaster was not available for a more detailed comparison, we expect a constraint based approach to be more efficient than a model elimination theorem prover.) Due to its use of *safe iff definitions* for representing sources (and domain knowledge), Infomaster employs *gensym predicates* for emulating unidirectional (necessary) source definitions (which represent the typical case, since only very rarely do we have *complete* source descriptions).

COIN [BG97] also uses a CLP framework (Eclipse) for abductive reasoning and CHRs for implementing integrity constraints. However, integrity constraints can be imposed in COIN *only on source predicates*. Thus, COIN domain knowledge reduces to Prolog reduction rules, which are used *only backwards* (during goal regression). The lack of forward propagation rules involving base predicates (and not just sources) makes the discovery of potential interactions between base predicates (and thus the full use of domain knowledge) impossible. COIN also doesn't have a full-fledged query planner (it doesn't aggregate sub-queries to relations of the same source, thereby having to transport large/huge numbers of tuples from the source to the mediator).

CHR and its disjunctive extension CHR$^{\vee}$ [AS98] have also been used by Abdennadher and Christiansen [AS00] as a platform for integrity constraints and abduction.

9 The interactions between sources from different "buckets" (in the terminology of [LRO96]) need to be taken into account.

10 Consistency tests in a sophisticated language like CARIN [LR98] being "lazy" (a complex external consistency verifier for CARIN is invoked).

However, CHR^{V} and SLPs [KTW98] and represent more general architectures (very much like CHRs), while we are concentrating on query planning in the framework of intelligent information integration. (While the criterion for goal suspension in SLPs is the instantiation of variables, in query planning we have a more sophisticated criterion stopping unfolding at source predicates, aggregating them into source queries and instantiating variables by executing these partial queries.) We also propose a higher level knowledge representation formalism based on description rules and use CHRs as a means of combining backward and forward reasoning in this formalism.

We are unaware of any I^3 system based on SLPs or CHRs.

In a recent paper [GM02], Grant and Minker present an elegant logic-based approach to data integration. However, Grant and Minker deal with the more difficult problem of answering queries using views (which is more appropriate for a data warehousing approach to integration), while we are more interested in what has been recently called 'real-time data integration' (which is more appropriate whenever the sources change frequently so that data warehousing is inapplicable). The state of art on answering queries using views is reviewed in [H01], which compares three algorithms, namely the bucket algorithm [LRO96], inverse rules [DG97a] and the MiniCon algorithm [PL01], showing that the MiniCon outperforms the other two.

Finally, Denecker et al. [D95] use the notion of "open predicates" as a synonym of abducibles, or of what we call "completely open" predicates. We generalize abducibles by allowing open predicates to have partial definitions.

The system presented in this paper (implemented in SICStus CHR [Sics]) is fully operational and has been used in several real-world applications involving the integration of molecular biology and genetics resources (databases, knowledge bases and tools), as well as corporate information systems.

Acknowledgments. This work was partially supported by the European Commission IST project SILK (IST-1999-11135) [S]. We are grateful to our project partners IQSOFT (Hungary) and EADS (France) for developing the database wrappers and the Sicstus Prolog interface to them, as well as for interesting discussions on the topics presented in this paper. The SILK toolset developed in the framework of the abovementioned project is a more complex environment including a mediator (presented in this paper), but also a so-called 'integrator' (used for model construction and evolution) [BKS02], as well as various wrappers and a developer interface.

References

[AART97] Aquilino D., Asirelli P., Renso C., Turini F., MedLan: a Logic-based Mediator Language, IEI Technical Report B4-16, November 1997.

[AC00] Abdennadher S., Christiansen H. An Experimental CLP Platform for Integrity Constraints and Abduction. Proc. FQAS-2000.

[AKH96] Y. Arens, C.A. Knoblock, Chun-Nan Hsu. Query Processing in the SIMS Information Mediator, Advanced Planning Technology, A.Tate (ed)), AAAI Press, 1996.

[AS98] Abdennadher S., H. Schuetz. CHR^{V}: a flexible query language. Proc. FQAS-98.

[BG97] S. Bressan and C.H. Goh. Answering queries in context. In Proceedings of the International Conference on Flexible Query Answering Systems, FQAS-98, Roskilde, 1998.

[BKS02] T. Benko, P. Krauth, P. Szeredi. Application Integration through Logic based Model Evolution. Proc. ICLP-2002.

[D95] M. Denecker. A Terminological Interpretation of (Abductive) Logic Programming. Proc. NMR-95, pp.15-29, 1995.

[DG97b] O.M. Duschka and Michael R. Genesereth. Query Planning in Infomaster, Proc.12th Annual ACM Symposium on Applied Computing, SAC '97, San Jose, February 1997.

[DG97a] O.M. Duschka, M. Genesereth. Answering recursive queries using views. PODS-97.

[DG97] O.M. Duschka, M.R. Genesereth. Infomaster - An Information Integration Tool. Tool. Proc. International Workshop "Intelligent Information Integration", KI-97, Freiburg, 1997.

[Fr98] Fruewirth T. Theory and Practice of Constraint Handling Rules, JLP 37:95-138, 1998.

[GM02 J. Grant, J. Minker. A logic-based approach to data integration. TPLP 2002.

[GPQ97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, J. Widom. The TSIMMIS approach to mediation: Data models and Languages. In Journal of Intelligent Information Systems, 1997.

[H01] Alon Halevy. Answering queries using views: a survey. VLDB Journal 2001.

[KKT98] Kakas A., Kowalski R., Toni F. The role of abduction in logic programming, Handbook of logic in AI and LP 5, OUP 1998, 235-324.

[KTW98] Kowalski R., Toni F., Wetzel G. Executing suspended logic programs, Fundamenta Informaticae 34 (1998), 1-22.

[KTW98] R. Kowalski, F. Toni, and G. Wetzel. Executing suspended logic programs. Fundamenta Informaticae, 34(3):203-224, 1998.

[Lev00] Alon Y. Levy , Logic-Based Techniques, in Data Integration Logic Based Artificial Intelligence, Jack Minker (ed). Kluwer, 2000.

[LRO96] Alon Y. Levy, A. Rajaraman, J.J. Ordille, Querying Heterogeneous Information Sources Using Source. Proc. 22nd VLDB Conference, Bombay, India. 1996.

[PL01] Rachel Pottinger , Alon Y. Halevy , Minicon: A Scalable Algorithm for Answering Queries Using Views VLDB Journal 2001.

[Sics] SICS. *SICStus Prolog Manual*, April 2001.

[Sub] V.S. Subrahmanian et al. HERMES: A heterogeneous reasoning and mediator system. http://www.cs.umd.edu/projects/hermes/overview/paper.

[S] The SILK project (IST-11135). http://www.silk-project.com/

[Wie92] Wiederhold G. Mediators in the architecture of future information systems, IEEE Comp. 25(3) 1992, 38-49.

[WT98] G. Wetzel and F. Toni. Semantic query optimization through abduction and constraint handling. Proc. FQAS-98 Flexible Query Answering Systems, LNAI 1495:366-381, 1998.

[W97] Wetzel G. Using integrity constraints as deletion rules, Proc. DYNAMICS'97, 147-161.

Effect Preservation as a Means for Achieving Update Consistency*

Mira Balaban[1] and Steffen Jurk[2]

[1] Ben-Gurion University
Beer-Sheva, Israel
mira@cs.bgu.ac.il

[2] Brandenburg Technical University
Cottbus, Germany
sj@informatik.tu-cottbus.de

Abstract. Management of data systems must cope with changes, initiated by users or applications. Query answering in a frequently modified data system must be consistent with the updates. A natural expectation is that a fact that was successfully added, is retrievable as long as it was not intentionally removed. Active Databases do not meet this expectation, since contradicting rules may undo events that triggered them. In this paper, we associate database transactions with effects, and present a method that takes care of preserving the effects of updates. We introduce a compile-time effect preservation transformation that revises a transaction so to prevent contradictory updates, i.e., yields an effect preserving transaction. Our method yields an expressive and efficient transaction, since it is based on interleaving run-time sensitive analysis of effects within the compile-time transformation of a transaction. The interleaving and the compile-time reduction of effects account for the efficiency; the run-time sensitivity of effects accounts for the expressiveness.
In the context of Active Databases this method can be used to statically revise a rule application plan, so to prevent contradictory updates. We claim that an effect preserving method should be integrated into every update and query processing system.

1 Introduction

Management of data systems must cope with changes, initiated by users or applications. Query answering in a frequently modified data system must be consistent with the updates. A natural expectation is that a fact that was successfully added, is retrievable as long as it was not intentionally removed. Such behavior is achievable if contradictory updates that undo each other are avoided. Active databases ([16,13,17]) do not meet this expectation since contradicting

* This work was supported in part by the Paul Ivanir Center for Robotics and Production Management at Ben-Gurion University of the Negev.
This research was supported by the DFG, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

rules may undo events that triggered them. Moreover, in a distributed active database a user might not be aware of rules that trigger contradicting actions, since the rules might reside in independently developed sites.

Example 1. Consider a database that consists of multiple sites, holding information about two unary relations A and B – for example relations that hold various code numbers. Each site is enforcing its own constraint, and there is noway for a user to know all the constraints. For the sake of the example, consider constraints. One site enforces an exclusion constraint $A \cap B = \emptyset$ using the following rules:

$$\begin{aligned} R_1 &: \text{ON } \; insert(A, x) : \text{IF } x \in B \text{ THEN } delete(B, x) \\ R_2 &: \text{ON } \; insert(B, x) : \text{IF } x \in A \text{ THEN } delete(A, x) \end{aligned}$$

Another site requires that new values in A are reflected by a related value in B (for some function f):

$$R_3 : \text{ON } \; insert(A, x) : \text{IF } f(x) \notin B \text{ THEN } insert(B, f(x))$$

Assume that we need to add a security code to table A in order to grant access to a specific person. There are two requirements: (1) *state consistency*: The resulting state is consistent with respect to all site constraints. (2) *update consistency*: If the transaction terminates successfully, it should be reflected in successive query answering. That is, any query that is conditioned by the stored access code of that person should succeed.

In order to obtain state consistency, the rules of both sites have to be executed. Assume that rule execution is defined as follows: A rule is fired immediately AFTER the primitive update (insert or delete) causing that rule is executed, and if multiple rules are fired by a primitive, they are sequentially applied according to their internal ordering (e.g., in this example, R_1, R_2, R_3). Using this rule application policy, the primitive $insert(A, x)$ yields the following state consistent transaction:

```
$insert\_code\_to\_A(x) = insert(A, x);$
                          IF $x \in B$ THEN $delete(B, x)$;
                          IF $f(x) \notin B$ THEN $insert(B, f(x))$;
                                  IF $f(x) \in A$ THEN $delete(A, f(x))$;
```

This transaction can be triggered by applications of the update $insert(A, x)$, as a consistency preserving repair[1].

In order to obtain update consistency, any successfully added access code should be retrievable from A. This requirement is not achieved, for example, by the application $insert\_code\_to\_A(3)$ in a state with $A = \{1\}$, $B = \{2\}$, $f(3) = 3$, which yields:

[1] A similar approach is adopted in [6,7] for the enforcement of cardinality constraints.

Rule	Primitive Update	A	B
–	$insert(A, 3)$	$1, 3$	2
R_1	–	$1, 3$	2
R_3	$insert(B, 3)$	$1, 3$	$2, 3$
R_2	$delete(A, 3)$	1	$2, 3$

The above example is not update consistent since a seemingly successful code insertion ends up in a state where the person has no granted access. The update inconsistency is caused by allowing transactions that include contradictory updates, i.e., updates that undo the expected effects of previous updates, to succeed. The application of the transaction $insert\_code\_to\_A(3)$ to the state $A = \{1\}$, $B = \{2\}$, $f(3) = 3$, must fail (be rejected) since there is no way to achieve both state consistency and update consistency. Transactions that do not include contradictory updates are termed *effect preserving*.

In this paper we suggest a static analysis refinement approach for achieving both state consistency (as enforced by the rules) and update consistency in active database applications. The analysis includes the following steps:

1. **Rule Execution Plan**: For each primitive update, obtain a non-deterministic rule execution plan. A path in the plan reflects a concrete rule application, and yields a transaction.
2. **Effect Preservation Transformation**: Every transaction, derived from a path in a rule execution plan, is transformed into a new transaction that is effect preserving.

The analysis is restricted to active databases with rules that conform with the transaction language studied in this paper.

The effect preservation transformation is based on characterization of *contradictory updates*. The problem was first discussed in the early planning systems in Artificial intelligence. Later on it was studied within the context of integrity enforcement in databases, where repairing updates can undo each other ([3,4]). However, it is not handled in regular database maintenance systems like RTSs.

In order to keep our methods domain independent, i.e., applicable to different forms of data (not necessarily relational), we adopt an intention oriented approach. That is, contradictory updates are defined as updates that lead to contradiction of intentions, as in insertion or deletion of a single element or a link to database tables or web sites. In these cases it is easy to single out pairs of contradictory operations ([14,10]). However, when primitive operations can involve multiple elements, as in insertion or deletion of a group of tuples (e.g., following some selection), the notion of intention can vary.

It seems that the characterization of intentions of primitive updates should be left for the initiator to specify. This approach is in accordance with the common approach in operation specification in systems design and programming ([5,11]). In these paradigms the designer associates operations (events) with *contracts*, that specify, among other things, the *effects* (changes) caused by the operations.

We follow that approach for the task of avoiding contradictory updates. The effects of *primitive updates* are post-conditions that specify their *intentions*, and are set by the user (update initiator). The effects of *composite updates* (*transactions*) are computed by the update management system. Effects of updates are defined as constraints (logic formulae) that describe the intended semantics of the update, either statically or dynamically. That is, every update is associated with a constraint that captures its intention.

Our goal is to carefully repair transactions so that their intention is not changed, but computation paths that might lead to an unsatisfiable intention are *annotated* with necessary intention preserving tests. The two criteria for evaluating the result of the repairing transformation are: (1) Whether the output transaction leads too often to unnecessary rejections (signifying non-delicate characterization of effects); and (2) The additional run-time overhead imposed on the resulting output (effect preserving) transaction.

The major part of this paper devoted to the introduction of a compile-time *effect preserving* transformation of transactions, that relies on fine, run-time sensitive characterization of effects. The compile-time processing includes two steps:

1. Construction of a *computation tree*, whose nodes are labeled with maximally reduced constraints that are necessary for effect preservation.
2. Transformation of the given transaction into a new transaction in which tree navigation and constraint evaluations are embedded.

Splitting the load this way, the major part is performed at compile-time, while at run-time only the statically unresolved intentions are left for evaluation. Since the size of the reduced conditions is independent of the size of the transaction and is expected to be rather small and easy to evaluate, the run time overhead is linear in the size of the transaction. The novelty of our approach is in combining the compile-time and the run-time processing in a way that run-time overhead is minimized, without sacrificing the expressivity of the revised transaction.

In Section 2 the update language is defined, and Section 3 introduces the path sensitive characterization of effects (update intentions). Section 4 presents the combined static-dynamic revision of transactions, that enforces the effect preservation property. Section 5 concludes the paper.

2 A Restricted Language of Rules and Updates

The update language studied in this paper is restricted to an imperative language that with a sequence and a conditional update combinators. The updates are built over a finite set of typed *state variables* X (a *state space*). A *state* is a well typed value assignment to the variables in X. For example, in a relational database with relations $R_1, \dots, R_n$, the state space is $\{R_1, \dots, R_n\}$, and any assignment of concrete relations to the relation variables results a database state. A language over X is denoted $\mathcal{L}(X)$.

The language symbols include, besides the state variables, input and local variables, and self-evaluating symbols (language constants). Input variables are not assignable. The primitive update command are *skip* (a no-op operation), *fail* (rollback, the impossible update), and *assignment updates* – well typed assignments to state variables. Assignment updates include, besides the assigned state variable, only input variables and constants, i.e., non-assignable symbols. The *fail* operation leads to the *undefined state*, which in transactional databases correspond to the *rollback* operation, which undoes undefined states by restoring the old state.

The two constructors that are studied in this paper are sequential composition, denoted $(S_1; S_2)$ and guarded deterministic choice (conditional), denoted (`if` P `then` S_1 `else` S_2) where S_1 and S_2 are operations and P a condition. We use `if` P `then` S for abbreviating `if` P `then` S `else` *skip*. The formal syntax of the update language is defined in the full paper [9]. The formal semantics of the language is defined as in Dijkstra's guarded commands language ([2, 12,4]. An extension of the language is discussed in the conclusion section.

The rules are of the form: ON $update(x, f(\bar{e}))$: IF $p(\bar{e})$ THEN $S(\bar{e})$. where x is a state variable, $\bar{e}$ is a set of input variables, $f(\bar{e})$ an expression over $\bar{e}$, $p(\bar{e})$ is a condition (formula) whose free variables are either state variables or in $\bar{e}$, and $S(\bar{e})$ is a transaction in $\mathcal{L}(X)$, with input variables $\bar{e}$. That is, rule conditions and actions do not include input variables that do not appear in their triggering events.

As described in the introduction, every path in a rule execution plan of an assignment update determines a transaction. The application of the transaction is equal to the application of the assignment, followed by the triggered rules. The following algorithm derives a transaction under a given rule application ordering in a set of rules $\mathcal{R}$:

Algorithm 1 *The transaction derivation algorithm. Derive a transaction for a given assignment update event, and a given rule selection policy in a set of rules $\mathcal{R}$. In order to guarantee termination, we assume the existence of a* cycle bound. *Cycle management is omitted from the algorithm, see [15,1] for details.*

```
deriveTransaction($event$) =
  Let $R_1, \dots, R_n$ be the selected ordering of rules in $\mathcal{R}$ for which
  $event(R_i) = event$;
  For each $R_i \in \mathcal{R}$:
      $C_i = condition(R_i)$;
      if cycle bound is not exceeded:
        for each assignment update $U_j$ in $action(R_i)$:
          replace $U_j$ by deriveTransaction($U_j$);
      $A'_i = action(R_i)$, after the replacement of the assignment updates.
      $R'_i =$ IF $C_i$ THEN $A'_i$.
  Output: $event; R'_1; \dots; R'_n$
```

Transactions obtained in example 1 from the rules is derived with this algorithm.

Proposition 1. *The transactions obtained by Algorithm 1 are equivalent to rule applications under the selected rule application policy.*

In the rest of this paper we directly consider transactions, without mentioning the rules from which they are derived.

3 Path Sensitive Characterization of Effects

In this section we define the effects of composite transactions, on the basis of effects of primitive updates. As discussed in the introduction, effects of primitive updates reflect the intentions of the developer. First we define effects of sequences of primitive updates. We distinguish between *desired effects*, denoted $\mathit{effects}_D(S)$, to *executed effects*, denoted $\mathit{effects}_E(S)$: The first, expresses the aggregated desired effects of all primitive updates in a transaction and it might not hold after the transaction is completed. The second, expresses the historical effects of all primitive updates in a transaction, as they were when executed, and it always holds after a transaction is computed. Then, we introduce the notion of a *computation tree* associated with a transaction, and assign *guarded effects* of both kinds with its nodes. Finally, we define the desired/executed effects of a transaction as the conjunction of the guarded desired/executed effects of the leaves in its computation tree, respectively. The section ends with a formal definition of the *effect preservation* property of transactions.

3.1 Effects of Primitive Updates

Primitive updates are atomic elements of the language $\mathcal{L}(X)$. Their intentions, denoted $\mathit{effects}(S)$ express postconditions that should hold following the update. The desired/executed distinction does not apply to primitive updates, since their effects are both desired and executed. For the two primitives *skip* and $fail$, their effects derive from their intended semantics[2]: $\mathit{effects}(skip) = true$, $\mathit{effects}(fail) = false$. For assignments, the effects are domain specific and developer provided. Clearly, we expect that developer provided effects are non-trivial, e.g. $\mathit{effects}(S) = true$.

Example 2 (Possible effects of primitive assignments in different domains).

- **Sets** – insert or delete an element e from a list x: $\mathit{effects}(x := insert(x, e)) = (e \in x)$, and $\mathit{effects}(x := delete(x, e)) = (e \notin x)$.
- **Lists** – insert an nth element e to a list x: $\mathit{effects}(x := insert(x, n, e)) = (e = element(x, n))$.
- **Trees** – insert an element e to a tree x: $\mathit{effects}(x := insert(x, path, e)) = (e = element(x, path))$.

The effects in the last example involve conditions that can be expressed in terms of the final values of the state variables, i.e., static conditions. In this paper we deal only with static effects, and most of the examples are taken from the domain of sets.

[2] Recall that $fail$ is the impossible update, i.e., rejection. Its semantics in Dijkstra's guarded commands language [2,12] states that "everything holds following a $fail$".

3.2 Effects of Sequences of Primitive Updates

Primitive updates modify the values of state variables. These modifications have to be considered in the account of the desired and executed (historical) effects, because repetitive modifications of a state variable might interfere. Therefore, the computation of effects requires repeated application of the modifications to the state variables. The following examples demonstrate these notions in the set domain. They use the updates $x := \mathit{insert}(x, e)$ and $x := \mathit{delete}(x, e)$, for a set variable x and an element variable e, with the effect formulae $e \in x$ and $e \notin x$, respectively.

Example 3 (Desired Effects). Consider the transaction: $S(e_1, e_2, e_3) = \big(x := \mathit{insert}(x, e_1); x := \mathit{insert}(x, e_2); x := \mathit{delete}(x, e_3)\big)$. The desired effects are: $\mathit{effects}_D(S(e_1, e_2, e_3)) = e_1 \in x \wedge e_2 \in x \wedge e_3 \notin x$. Clearly, $\mathit{effects}_D(S(e_1, e_2, e_3))$ does not hold in case that $e_1 = e_3$ or $e_2 = e_3$.

Example 4 (Executed (Historical) Effects). Consider the transaction S above. The executed effects of S certainly include $e_3 \notin x$. The insertion of e_2 occurs before the last deletion. Therefore, in terms of the final value of x, its effect is $e_2 \in \mathit{insert}(x, e_3)$, i.e., the last *delete* update must be reversed. The insertion of e_1 occurs before the insertion of e_2. Therefore, in terms of the final value of x, its effect is $e_1 \in \mathit{delete}(\mathit{insert}(x, e_3), e_2)$. Altogether we have: $(e_1 \in \mathit{delete}(\mathit{insert}(x, e_3), e_2)) \wedge (e_2 \in \mathit{insert}(x, e_3)) \wedge (e_3 \notin x)$. Yet, these executed effects fall short of handling the case where $e_1 = e_2$. The problem has to do with idempotent updates (like set insertion and deletion), where repetitions are redundant since $((x \; op \; a) \; op \; a) = (x \; op \; a)$. If $e_1 = e_2$ the second insertion is a *skip* update, but in the executed effects account it is reversed by a non-*skip* update.

We can handle sequences of redundant primitive updates by factoring out repetitions. In this example, the second update $x := \mathit{insert}(x, e_2)$ is transformed into $x := \mathit{insert}(x, e_2 - e_1)$, where the element subtraction stands for singleton set subtraction. Therefore, the executed effects are: $\mathit{effects}_E(S(e_1, e_2, e_3)) = (e_1 \in \mathit{delete}(\mathit{insert}(x, e_3), e_2 - e_1)) \wedge (e_2 \in \mathit{insert}(x, e_3)) \wedge (e_3 \notin x)$

Definition of desired effects: For simple updates that involve a single state variable, the desired effects of a sequence are simply the conjunction of the effects of the primitive updates (as in the above example). For more involved updates consult [9].

Definition of executed effects: Let S be a sequence of primitive updates $x_1 := A_1, \ \ldots, \ x_n := A_n$. The executed effects of S are obtained in two steps:

1. **Factorization**: Each update $x_i := A_i$ is factored with respect to all previous updates $x_j := A_j$ where $x_i = x_j$ and $j < i$. The exact factorization procedure is type dependent.

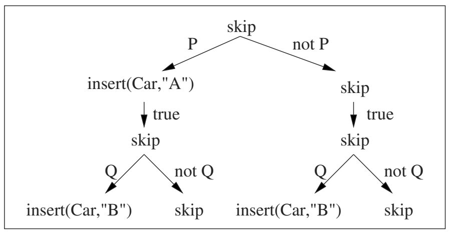
2. **Inductive definition**:
   a) For a primitive update U, $\mathit{effects}_E(U) = \mathit{effects}(U)$.
   b) – $\mathit{effects}_E(S;\ \mathit{skip}) = \mathit{effects}_E(S)$.
      – $\mathit{effects}_E(S;\ \mathit{fail}) = \mathit{false}$.
      – $\mathit{effects}_E(S;\ x := A) = \mathit{effects}(x := A) \wedge (\mathit{effects}_E(S)_{\{x/A^{-1}\}})$, with A^{-1} as inverse operation of A.

It can be shown that $\mathit{effects}_E(S)$ of sequences of primitive updates are valid postconditions of any execution of S. A formal proof requires introduction of a calculus for reasoning about imperative transactions, such as Dijkstra's guarded commands. The proof appears in the full paper [9].

3.3 Computation Paths and Effects

Compile-time effect preservation requires fine analysis of the computation paths of a transaction and their effects. The following examples demonstrate the weakness of effects that are assigned to a transaction as a whole, and the advantage of an effect preservation theory that relies on path sensitive effects.

Example 5 (Path Sensitive Effects).



The transaction

$S(\text{“A”}, \text{“B”}) =$
if P **then** $Car := insert(Car, \text{“A”})$;
if Q **then** $Car := insert(Car, \text{“B”})$

has four computation paths, with the desired effects: “A” $\in Car \wedge$ “B” $\in Car$, “A” $\in Car$, “B” $\in Car$, or $true$. (the latter effect means that only *skip* is performed). Each formula corresponds to the effects of a possible sequence of primitive updates.

In order to define effects on paths of a computation, we use an auxiliary data structure termed the *computation tree* of a transaction. Each transaction is associated with a single finite computation tree, whose paths span all the possible evaluations of the conditions in the transaction. The nodes of the computation tree are labeled with primitive updates, and the arcs are labeled by conditions.

Definition 1 (Computation Tree). Let $Node(U)$ be a constructor for a node tree labeled with a primitive update U, and $addLeft(node, P, T)$ and $addRight(node, P, T)$ methods that add a left or right child T to a node, with P as the arc condition. The computation tree $Tree(S)$ of a transaction S is inductively defined:

1. For a primitive update U, $Tree(U) = Node(U)$.
2. – $Tree(\ S_1; S_2\ )$ = for all leaves l of $Tree(S_1)$ do: $addLeft(l, true, Tree(S_2))$.

– $Tree($ `if` P `then` S_1 `else` S_2 $) =$
$root = Node(skip),\ \ addLeft(root, P, S_1),\ \ addRight(root, \neg P, S_2)$.

Each node in a computation tree stands for a sequence of primitive updates given by the labels on the path that leads to the node. Further, each node is associated with a *guard* constraint that characterizes the initial condition for following the path (a *pre-condition*).

Example 6 (Guards and Sequences of a Computation Tree Nodes). Consider example 5. The four leaves are associated with the following sequences and guards:

sequence	*guard*
$insert(Car, \text{"A"}); insert(Car, \text{"B"})$	$P \wedge Q_{\{Car/insert(Car, \text{"}A'')\}}$
$insert(Car, \text{"A"}); skip (\equiv insert(Car, \text{"A"}))$	$P \wedge \neg Q_{\{Car/insert(Car, \text{"A"})\}}$
$skip; insert(Car, \text{"B"}) (\equiv insert(Car, \text{"B"}))$	$\neg P \wedge Q$
$skip; skip (\equiv skip)$	$\neg P \wedge \neg Q$

Guards of nodes can be defined inductively, similarly to the definition of executed effects ([9]). Computation trees can be optimized by pruning branches that include internal $fail$ nodes, and removing nodes whose guards are equivalent to $false$.

Each node in a computation tree is associated with the executed and desired effects of its sequences. For example, the left most leaf in the computation tree in example 5 is associated with the desired effects "A" $\in Car \wedge$ "B" $\in Car$ and the executed effects "A" $\in delete(Car, \text{"B"}) \wedge$ "B" $\in Car$. The *guarded desired/executed effects* of a node n are the implication conditions $guard(n) \Rightarrow effects_D(n)$ and $guard(n) \Rightarrow effects_E(n)$, respectively.

3.4 The Effect Preservation Property

The effects of a general composite transaction should take into account its different computation paths. Otherwise, the effects impose restrictions that are not met by most transactions. For example, the transaction of example 5 cannot have as its effects simply the conjunction of the contradictory effects of its primitive updates. Rather, the effects of a composite transaction can use the guards on its computation tree to restrict the effects of its computation paths:

Definition 2 (The Desired/Executed Effects). *The* desired/executed effects *of a transaction S are given by the conjunction of the guarded desired/executed effects of the leaf nodes in the computation tree of S.*

Following the intuitive discussion of effect preservation above, we define a transaction as effect preserving if its desired effects follow from its executed effects. That is, the history that followed a modification did not affect its effect:

Definition 3 (The Effect Preservation Property). *A transaction S is* effect preserving *if $effects_E(S) \Rightarrow effects_D(S)$.*

Proposition 2. *A transaction is effect preserving if and only if for each leaf node n of its computation tree, $\mathit{effects}_E(n) \Rightarrow \mathit{effects}_D(S)$ is valid.*

Example 7. The transaction from example 5 is effect preserving, since it can be easily verified that for each leaf node the last proposition holds. For example, for the left most leaf, $[\text{“A”} \in \mathit{delete}(\mathit{Car}, \text{“B”}) \wedge \text{“B”} \in \mathit{Car}] \Rightarrow \text{“A”} \in \mathit{Car} \wedge \text{“B”} \in \mathit{Car}$ is valid.

4 The Effect Preservation Transformation

An effect preserving transformation turns an input transaction into an effect preserving one, by inserting tests that act as guards against violation of previous intentions. The two criteria for a good transformation are: (1) Minimize rejections caused by effect preservation and (2) Minimize the run-time overhead. The task of effect preservation can be carried out at run-time, or at compile-time, or in a combined mode. A run-time only transformation achieves the first criterion since rejections are triggered only in case of past effect violation. However, the run-time overhead is maximal, since at every modification all past effects must be checked. A compile-time only transformation achieves the second criterion since all effect violations are pre-computed and interleaved within the transaction, but the first criterion is not achieved since evaluation of effects requires run-time information.

In a combined compile-time run-time approach both criteria can be achieved using static analysis and partial evaluation of effects. The path sensitive effects characterized in the previous section guarantee that tested effects reflect the intentions of updates that are actually executed, and do not enforce effects of computation paths that are not followed. The static analysis in the presented transformation includes the construction of the computation tree with its path sensitive effects. Furthermore, in each tree node, the relevance between the aggregated path effects and the immediate primitive modification is pre-computed. This relevance singles out *delta-conditions* that must be checked, following the previous tests. This way, repeated time consuming evaluations of full path effects before or after modifications are saved. The delta-conditions provide a form of partial static evaluation of effects. The final evaluation of the delta-conditions is left to run-time. Run-time overhead is minimized by interleaving the tree navigation within the resulting transaction, and computation of delta-conditions in compile-time.

First, we introduce a naive effect-preserving transformation that takes care of the first criterion by using the path sensitive effects. Then, we provide a run-time improvement by interleaving navigation with performance, introduction of delta-conditions, and applying tests before modifications.

4.1 Naive Effect Preservation – Exhaustive Postconditions

The main idea is to enforce effect preservation following every assignment. We start using a “post-condition” because the effects that are associated with the nodes of a computation tree express the intentions following a transaction. In order to test whether a primitive update would violate the aggregated path effects before it is actually applied, the effects should be worked out into preconditions. This is indeed done within the computation of the delta-conditions below.

The *Naive Effect Preservation* algorithm inserts an effects test after every assignment. That is, an assignment A is replaced by: $A;$ `if` $\neg P_A$ `then` $fail$, where P_A is the condition that tests whether A preserves the effects of previously executed updates. This way, computation paths that lead to non effect preserving updates are rejected. The condition P_A is compile-time defined using the tree nodes that correspond to A (denoted $nodes(A)$, their guards and desired effects. Each node n in $nodes(A)$ represents a single computation path that might lead to execution of A. Its guard, $guard(n)$, is a pre-condition for following the path, and its desired effects, $effects_D(n)$, is the path effect. A fine and sensitive definition of P_A is obtained by first determining the single computation path that is being followed. This is done by preceding the evaluation of the effects of a node with its guard evaluation. The resulting P_A condition is: $\bigwedge_{n \in nodes(A)} (guard(n) \Rightarrow effects_D(n))$.

Algorithm 2 (Naive Effect Preservation)

```
naiveReviseUpdate($S$) =
  Compute Tree ($S$);
  For each assignment $A$ in $S$:
    compute: $P_A = \bigwedge_{n \in nodes(A)} (guard(n) \Rightarrow effects_D(n))$.
    replace $A$ by: ($A$;if $\neg P_A$ then $fail$)
```

Proposition 3 (Correctness of Algorithm 2). *Algorithm 2 yields an effect preserving transaction.*

Algorithm 2 minimizes rejections since it enforces only effects of executed computation paths. However, although the computation tree and its guards and effects are pre-computed, the evaluation of P_A at run-time is still costly, since it implies multiple evaluations of guards and effects for all computation paths where A occurs, in order to determine the actually executed path.

4.2 Efficient Effect Preservation – Reducing Run-Time Overhead

Observing the naive algorithm, it becomes clear that associating every assignment with tests for the single executed computation path is redundant since the transaction itself already has done it. Therefore, if navigation over the pre-computed tree is interleaved within the transaction, the run-time overhead of path determination (evaluation of $guard(n)$ for all nodes n in $nodes(A)$) is saved.

For each assignment it is left only to check the condition $effects_D(n)$. However, this is still costly, since the size of $effects_D(n)$ is proportional to the size of the transaction. Therefore, the run-time overhead is in worst case order of

$\mathcal{O}(size(S)^2)$, for a transaction S, assuming that primitive effect tests take constant time, and that primitive updates have atomic effects. Unfortunately, this is the run-time overhead of run-time effect preservation. The run-time overhead can be improved by minimizing the effects tests at every modification. This is done by associating tree nodes with *delta-conditions* that specify the past effects that might be affected by the next assignment.

Delta-conditions extract the possible interaction between aggregated path effects and an immediate primitive update. Consider, for example, the transaction $S(u, v) = (x := insert(x, u); x := delete(x, v))$. The desired effects along its single path computation tree are: *true*, $u \in x$ and $u \in x \wedge v \notin x$. Again, static analysis of these effects leaves the delta-condition $u \neq v$ to be tested prior to the application of the second assignment.

Definition 4 (Delta Conditions). *Let n be an assignment node in a computation tree of a transaction, with an update label $x := f(\bar{e})$. If n is the root node, then $delta\_condition(n) = true$. Otherwise, $delta\_condition(n)$ is any condition satisfying: $effects_D(parent(n)) \wedge delta\_condition(n) \Rightarrow effects_D(parent(n))_{\{x/f(\bar{e})\}}$*

That is, the delta condition gurantees that the desired past effects of the parent node hold after the assignment. Clearly, $effects_D(parent(n))_{\{x/f(\bar{e})\}}$ is a legal $delta\_condition(n)$, but the worst. The best $delta\_condition(n)$ is simply *true*, which indeed is the case, if the update of n does not interfere with $effects_D(parent(n))$.

Delta-conditions express the difference between the desired past effects of a child node to those of its parent node. They can be computed by a rough syntactic method that just considers common occurrences of variables, or by a fine, domain dependent method, that exploits knowledge about the domain primitives, i.e., transactions and predicates. For example, given the transaction $S(u, v) = (x := insert(x, u); y := delete(y, v))$. A syntactic analysis can associate the delta-condition *true* with the tree nodes of both assignments, based on a “no common state variables” consideration.

The computation tree of a transaction is extended with delta-conditions. A delta-condition is computed between a primitive assignment and its past effects. For each non-root primitive assignment node n, with $update(n)$ and parent node $parent(n)$ we compute $delta\text{-}condition(update(n), effects_D(parent(n)))$ and associate it with the node n. That is, $delta\text{-}condition(n)$ provides the minimal conditions that should be tested for the effect preservation of $update(n)$. Moreover, the delta-condition can be tested prior to the application of the update. This is important, in particular, for applications where the primitives might have side effects that might not be reversed by a mere rejection of the transaction. Examples of such systems are reactive systems, where updates trigger immediate actions.

A delta-condition specifies a test for possible interference between a transaction and a constraint. Static evaluation of delta-conditions can be performed with methods developed in the field of *Integrity Constraint Checking* [14,10], where static analysis results minimal conditions that are left to be tested at run-time.

Algorithm 3 (Effect-Preservation with Delta-Conditions)
The pre-computed tree is extended with delta-conditions. The algorithm uses an auxiliary `reviseUpdate` *syntactic transformation, that interleaves the tree navigation (move to left or right child node) using the ref variable, and refers to the delta-conditions.*

```
EP($S$) =
  $T$ = extend-with-delta-conditions(Tree($S$));
  $S'$ = reviseUpdate($S$, $T$);
  Output: $(ref := T;\ S')$

reviseUpdate( $S$, $T$ ) =
  replace each [if $P$ then $S_1$ else $S_2$] in $S$ by:
    (if $P$ then $ref$:=left($ref$);$S_1$ else $ref$:=right($ref$);$S_2$)
  for each primitive $U$ in $S$
    if $U = skip$ or $U = fail$ then replace $U$ by: ($U$;$ref$:=left($ref$))
    else
      if for all $n \in nodes(U)$ $delta\text{-}condition(n) = true$ then
        replace $U$ by: ($U$; $ref$:=left($ref$))
      else
        replace $U$ by:
        (if $delta\text{-}condition(ref)$ then $U$ else $fail$; $ref$:=left($ref$))
```

Algorithm 3 includes an additional static optimization step aimed at saving redundant tests of delta-conditions. Clearly, such tests at run-time are needed only if there exists at least one execution path in $nodes(A)$ with a delta-condition different from *true*. This optimization is especially important since we expect that effect violations would be rather rare, and the more common situation is that delta-conditions reduce to *true* at compile-time.

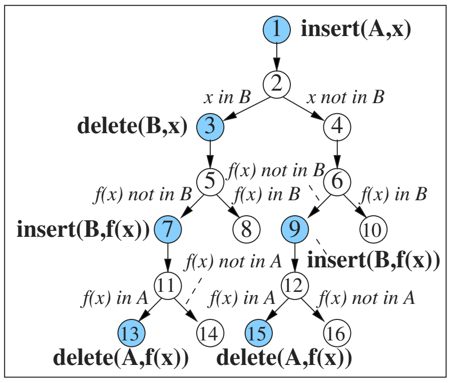
Finally, we present the main result showing that effect preservation is indeed achieved by the above algorithm, with run-time overhead that depends only on the delta-conditions. Since we believe that delta-conditions tend to be small and do not depend of the overall transaction size, then the run-time overhead is linear in the size of the transaction, instead of a quadratic overhead, in fully run-time effect preservation.

Proposition 4 (Correctness and Efficiency of the Algorithm). `EP` *is an effect preserving transformation. That is, for a transaction S in $\mathcal{L}(X)$,* `EP`(S) *is an effect preserving transaction. The expected run-time overhead of* `EP`(S) *over S is smaller than $size(S) \times size(delta\text{-}condition)$, i.e., $\mathcal{O}(size(S))$.*

4.3 An Example of the Effect Preservation Transformations

Finally, we demonstrate the transformation of a non effect preserving transaction into an effect preserving one, based on the rules given in example 1. Consider the transaction $S(x)$ that results from executing the rules R_1, R_2 and R_3 and the computation tree of S termed $Tree(S)$.

```
$S(x) = insert(A, x);$
        IF $x \in B$ THEN
            $delete(B, x);$
        IF $f(x) \notin B$ THEN
            $insert(B, f(x));$
            IF $f(x) \in A$ THEN
                $delete(A, f(x));$
```



The computation tree of $S(x)$. Dark nodes indicate assignments and light one skip primitives. Italic formulae denote conditions assigned to arcs.

Clearly, paths of the tree ending with the nodes 13, 14 and 15 might contain contradictory updates. The desired effects and delta-conditions of assignment nodes are depicted in the enclosed table. The delta-condition of node 1 reduces to *true*, since there is no previous update whose effects could be violated.

node	*effects$_D$*	*delta-condition*
1	$x \in A$	*true*
3	$x \in A \wedge x \notin B$	*true*
7	$x \in A \wedge x \notin B \wedge f(x) \in B$	$x \neq f(x)$
9	$x \in A \wedge f(x) \in B$	*true*
13	$x \in A \wedge x \notin B \wedge f(x) \in B \wedge f(x) \notin A$	$x \neq f(x)$
15	$x \in A \wedge f(x) \in B \wedge f(x) \notin A$	$x \neq f(x)$

The delta-conditions of the nodes 3, 9 also reduce to *true*, since the involved assignments affect different relations. Nodes 7, 13, 15 yield a delta-condition $x \neq f(x)$ which indicates, that if $x \neq f(x)$ does not hold, the effects are not satisfied (e.g. $x \in A \wedge x \notin A$). Applying Algorithm `EP` to S returns the following transaction:

```
      $S'(x) = ref := Tree(S);$
              $insert(A, x); ref := left(ref);$
              IF $x \in B$ THEN $ref := left(ref);$
                  $delete(B, x); ref := left(ref);$
              ELSE $ref := right(ref); ref := left(ref);$
              IF $f(x) \notin B$ THEN $ref := left(ref);$
                  IF $delta(ref)$ THEN $insert(B, f(x)); ref := left(ref);$
(*)               ELSE $fail;$
                  IF $f(x) \in A$ THEN $ref := left(ref);$
                      IF $delta(ref)$ THEN $delete(A, f(x)); ref := left(ref);$
(*)                   ELSE $fail;$
                  ELSE $ref := right(ref); ref := left(ref);$
              ELSE $ref := right(ref); ref := left(ref);$
```

Lines marked by $(*)$ denote positions where the transaction S' can be rejected due to the violation of effects. Consider again the state with $A = \{1\}$, $B = \{2\}$ and $f(3) = 3$. Unlike S which successfully terminates without achieving the intention of the triggering update, the execution of S' is rejected (no state change) due to the violation of effects:

Assignment Node	Delta-Conditon	Primitive Update	A	B
1	*true*	$insert(A, 3)$	$1, 3$	2
9	*true*	$insert(B, 3)$	$1, 3$	$2, 3$
15	$3 = f(3)$	*fail*	$1, 3$	2

5 Conclusion and Future Work

In this paper we defined the problem of user dependent effect preservation, and provided an efficient effect preservation algorithm with a linear run-time overhead. The algorithm relies on an exhaustive static analysis of a transaction, and on constraint reduction techniques. The static analysis includes path sensitive construction of effects, and reduction of interference conditions (delta-conditions). The resulting transformation is as expressive as the original transaction, but prevents effect violation.

We have shown how this method can be used to achieve update consistency, which is essential for having a faithful query system. The method is applicable to transactions that derive from stored procedures or user initiated. In addition we emphasized the relevance of update consistency in a distributed active database, where the actions that are triggered by the rules cannot be coordinated. For that purpose, we also introduced a method for deriving transactions from triggered rules (the rule execution plan). We are well aware that in some DBMS vendors like Oracle and DB2, rule application policy is determined at run time. For these applications, we have to further refine the transaction derivation procedure. Yet, the procedure fits vendors like SYBASE, where the order of rule application is fixed in advance.

The language considered in the paper is restricted to sequencing and conditional composite transactions alone, and the rule language is appropriately restricted. In order to strengthen the approach, it is necessary to extend the transformation to apply to a more expressive language of transactions. In particular, it is important to add bounded loops to the language.

In the future we plan to strengthen the update language, remove the restrictions on rules, and further investigate modes for supporting different execution orders of rules (e.g. simultaneous rules). Finally, we intend to construct a generic open tool for effect preservation that can be extended in terms of language and effects, and can be embedded in real active database systems.

References

1. E. Baralis and J. Widom. An algebraic approach to static analysis of active database rules. In *ACM Transactions on Database Systems*, volume 25(3), pages 269–332, September 2000.
2. E.W. Dijkstra and C.S. Scholten. Predicate calculus and program semantics. *Springer-Verlag, Texts and Monographs in Computer Science*, 1989.
3. K.D. Schewe and B. Thalheim. Consistency Enforcement in Active Databases. pages 71–76. IEEE Computer Society Press, 1994.
4. K.D. Schewe and B. Thalheim. Limitations of Rule Triggering Systems for Integrity Maintenance in the context of Transition Specifications. *Acta Cybernetica*, 1998.
5. C. Larmann. *Applying UML and Patterns.* Prentice-Hall, 1998.
6. M. Balaban and P. Shoval. Enhancing the ER model with structure methods. *Journal of Database Management*, 10(4), 1999.
7. M. Balaban and P. Shoval. MEER – an EER model enhanced with structure methods. *Information Systems Journal*, 10(4):245–275, 2002.
8. M. Balaban, S. Jurk. Improving Integrity Constraint Enforcement by Extended Rules and Dependency Graphs. In *Proc. 22th Conf. on DEXA*, 2001.
9. M. Balaban, S. Jurk. Intention of Updates - Characterization and Preservation. Technical report, Ben-Gurion University, Israel and BTU Cottbus, Germany, 2002.
10. F. Bry. Intensional updates: Abduction via deduction. In *Proc. 7th Conf. on Logi Programming*, 1990.
11. B. Meyer. *Object-Oriented Software Construction.* Prentice-Hall, 1997.
12. Greg Nelson. A generalization of dijkstras calculus. *ACM Transactions on Programming Languages and Systems*, 11:517–561, 1989.
13. P. Fraternali and S. Paraboschi and L. Tanca. Automatic Rule Generation for Constraints Enforcement in Active Databases. Springer WICS, 1993.
14. S.Y. Lee, T.W. Ling. Further Improvement on Integrity Constraint Checking for Stratisfiable Deductive Databases. In *Proc. 22th Conf. on VLDB*, 1996.
15. van der Voort and A. Siebes. Termination and confluence of rule execution. In *In Proceedings of the Second International Conference on Information and Knowledge Management*, November 1993.
16. J. Widom and S. Ceri. Deriving production rules for constraint maintenance. In *Proc. 16th Conf. on VLDB*, pages 566–577, 1990.
17. J. Widom and S. Ceri. *Active Database Systems.* Morgan-Kaufmann, 1996.

Recursive Queries in Product Databases

J.H. ter Bekke and J.A. Bakker

Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
{j.h.terbekke, j.a.bakker}@its.tudelft.nl

Abstract. The relational data model does not offer query language solutions for recursive applications. A missing semantic update operation is the fundamental reason for this failure. It is shown that a semantic data model can solve these applications using simple linear constructs for definition and manipulation, without explicit recursion, nesting, iteration or navigation. The new implementation technique is reliable and efficient for application in end user environments. It is based on semantic query specification and metadata processing. Experiments with the semantic Xplain DBMS have confirmed the linear time complexity for product database applications.

1 Introduction

The relational data model has had an enormous influence on the developments of information systems. With it, end user computing has settled down in the field of computer science. The relational data model with only linear structures replaced data models with nested data structures (such as hierarchical and network models). Consequently programming and navigation could be replaced by (interactive) associative access using SQL. In spite of its merits, this query language has still some shortcomings for advanced practical applications.

This paper is about one of the major flaws, namely the problem of data driven recursion in databases. This problem provided already in the 1960s a major impetus for the development of database technology [5]. As discussed before [12], SQL3 allows only logical deduction but not recursive calculation of aggregated data. The fact that relational languages cannot be used to express such problems [8] can therefore be considered as a major lack.

Several years of database research at TU Delft resulted in a simple semantic data model including query language (see [9] for an extensive motivation, formal definition and several applications) in which a number of shortcomings of present relational languages were tackled. In particular the following characteristics hold for this semantic approach as implemented in the Xplain DBMS:

- ***Reliability***
  The Xplain query language [9, 11] does not allow data manipulation constructs conflicting with the semantic constraints in a data model; it eliminates the majority of pitfalls that occur in user applications. Absence of an explicit join-operation avoids pitfalls caused by an empty set and also prevents to create

meaningless relationships. The Xplain language enforces to use only existing and meaningful paths [12]. This property is also indispensable for recursive applications.

- ***Predictability***
  It can be formally proven and it is in practice confirmed by Xplain DBMS that all queries in a semantic DBMS can be executed extremely fast in linear processing time, while higher-order and infinite processes (among others NP-complete problems) cannot be specified. The research prototype has therefore only an interactive user interface without batch processing (see also [10]). All user operations (manipulation and definition) can be performed on the fly, without unloading and reloading databases. Users can be enabled to work only on an interactive basis because processing time is predictable and because end users cannot specify queries requiring too many resources.
- ***Applicability***
  The language allows specification of a large class of practical problems. This class includes all queries specifiable according the semantic properties of a data model. The language is in that sense also relational complete. Moreover this class includes recursive queries that are inexpressible in a relational query language, such as critical path [13] and product database applications.

Because database queries should be executable in acceptable time, it is reasonable that a query language should not allow end users to formulate NP-hard problems. This is true for the Xplain query language: the language prevents users from formulating intractable problems [4] such as the traveling salesman problem or unsolvable problems [3] such as the halting problem. This restriction is inherent in the syntax of the language and its semantics. This is during run time enforced by the parser of Xplain DBMS in which nothing is compiled and everything is done by interpretation using metadata in an active data dictionary. This property offers also new perspectives in an Internet environment [2].

There is another proposal [7] for a functional query language that allows formulations of product database queries in polynomial execution time. This is considered as a theoretical achievement because the approach is not really practical in an environment of end user computing: recursive functions are too complex, it does not lead to sufficient metadata to tune the system processes for efficiency, it relies completely on the end user and the approach is inapplicable to large databases.

The exact determination of the collection of relevant semantic constraints appeared to be the most difficult part in the research; especially the mental shift from complete user control over recursion to complete release of this aspect was essential for a breakthrough in thinking. A new balance had to be found between the capabilities of end users and the capabilities of the system's parser with other algorithms. This balance resulted in user-friendly solutions with efficient and reliable implementations in Xplain DBMS.

The paper starts in section 2 with the fundamental data modeling concepts for recursive applications, such as product database applications. Similar semantic constructs are also used in other approaches but it is important to recognize that they have here their own concise definitions. This summary clarifies the small but crucial differences with relational structures (see also [6]). In section 3 some recursive applications are solved using the Xplain query language. The linear constructs in this language do not contain explicit recursion, nesting, iteration or navigation. Section 4 contains a global overview of the implementation technique that enabled us to process all queries in linear processing time. Test results on various databases of different sizes confirm the linearity of the processing time of the presented queries.

2 Semantic Model

A product database plays a role in any manufacturing facility. Each finished product recursively consists of one or more components until the basic components of the finished product are reached. The product structure denotes the hierarchy (generally and ideally not a tree) of the relationships between parts. This hierarchy will formally be defined below.

Each object in a semantic model is visualized by clearly distinguishing between identification and description. The resulting data models gain in semantic contents as a consequence, while ambiguities and contradictions in the specification are avoided. The abstractions classification, aggregation and generalization are considered here. They have clear graphical equivalencies in the structural diagrams and are based on the fundamental *type-attribute relationship*.

Classification is defined as the abstraction leading to a type. The examples (i.e. instances) occurring in a database support the recognition of a type but are purely illustrations of the concept. A type is *not* defined by its instances; query operations do therefore not affect the definition of a type. Types are represented by rectangles in diagrams, see figure 1. The counterpart of classification is called *instantiation*.

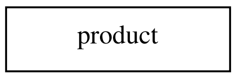


Fig. 1. Classification

Aggregation is defined as the collection of a certain number of types in a unit, which in itself can be regarded as a new type. A type occurring in an aggregation is called an attribute of the new type. It is important to note the analogy with the mathematical set concept: attributes can be considered as the 'elements' of a type.

Aggregation allows view independence: we can discuss the obtained type (possibly as a property) without referring to the underlying attributes. By applying this principle repeatedly, a hierarchy of types can be set up. An example of such a hierarchy is a data model for a directed graph (figure 2) in which product is corresponding with the node and fraction with the edge.

A line connecting two facing rectangle sides, while the aggregate type is (according to its definition) placed above its attributes, indicates aggregation.
Of course, aggregation also has its counterpart: the description of a type as a set of attributes is called *decomposition*.

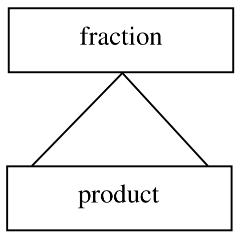


Fig. 2. Aggregation hierarchy

A type is completely defined by listing its attributes, so we could have the following type definitions in the product database:

type product	= name, stock, ordered_quantity.
type fraction	= major_product, minor_product, multiplicity.

An example to illustrate the corresponding database contents is table 1.

Table 1. Example database

product	name	stock	o_qty
p1	television	125	300
p2	tuner	35	25
p3	video player	12	220
p4	tube	100	45
p5	receiver	85	25
p6	transformer	235	100

fraction	major_p	minor_p	mult
f1	p1	p2	1
f2	p1	p4	1
f3	p1	p5	1
f4	p1	p6	1
f5	p3	p5	1
f6	p3	p6	2

Type definitions carry inherent semantics; they contain the essential properties (e.g. uniqueness of the identifications p1, p2, etc. in the product table and f1, f2 etc. in the fraction table) and essential relationships (e.g. related products p1, p2, p4, etc. of the fraction table must occur in the related product table). Aggregation can be described using the verb *to have*. According to the above type definitions, a product has a name, stock and ordered_quantity, and a fraction has a major_product, a minor_product and a multiplicity in which the minor_product occurs in the major_product. Identifications are denoted by type names (see table 1 above). This interpretation implies singular identifications for instances and types. Attributes (not types!) may contain *roles*; an example is ordered_quantity related to base type quantity. Attributes major_product and minor_product are related to type product. Roles are separated from the type by an underscore. The expression 'A *its* B' is meaningful by definition, it denotes the attribute B of type A. In case type B has an attribute C, we may also use the expression 'A *its* B *its* C'. Only this semantic *its*-construct can be used in attribute expressions next to simple variables and constants.

The third kind of data abstraction, important to conceptual data models, is called *generalization*; it is defined here as the recognition of similar attributes from various types and combining these in a new type (note again the analogy with the intersection operation from mathematical set theory). We can equally discuss the new type without mentioning the underlying attributes, and it can in itself again serve as a property (i.e. it allows view independence). For example we could consider the type product also as a generalization of the types 'mechanical product' and 'electronic product'. Generalizations can be represented in abstraction hierarchies, see figure 3.

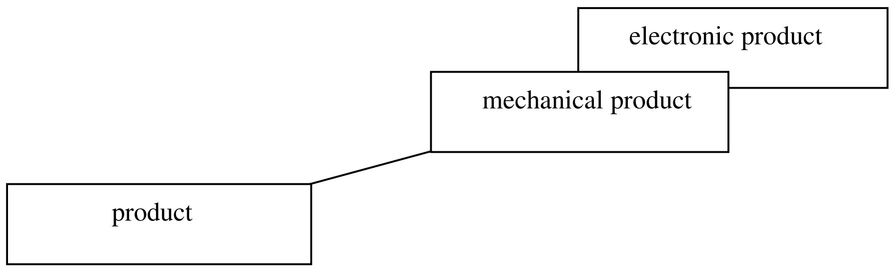


Fig. 3. Generalization hierarchy

In abstraction hierarchies, generalizations are schematically represented by a line connecting facing corners of rectangles, the generalized type being placed below the specialized ones (again according to its formal definition). Generalization's counterpart (i.e. the union of attributes from different types) is called *specialization*. Specializations can be described using the verb *to be*, for example: an electronic product is a product, has voltage, and has a current. This is reflected in following formal definitions:

type product = name, stock, ordered_quantity.
type electronic product = [product], voltage, current.
type mechanical product = [product], fuel, power.

In figure 3 we placed the types mechanical product and electronic product one behind the other, while only one line connects to type product. This is the usual representation of disjoint specializations: i.e. a product can be either mechanical or electronic, but not both. The combination of a group of disjoint specializations is called a *block*, so the types mechanical product and electronic product constitute a block. Types in a block contain the same referencing attribute between brackets. Although generalization can play a role in product databases, it is not discussed here because the querying of generalization (and specialization) is already covered by other publications (e.g. in [9]) and not really contributes to a demonstration of the query capabilities in the present context.

The 'vertical' orientation for placement of aggregation and generalization in abstraction hierarchies improves comprehensibility and facilitates data manipulation. This orientation reduces the number of possible graphical presentations. In other ER-related models (such as UML) structure is specified through cardinalities. They allow a large number of different diagrammatic presentations for one single problem, hereby hindering reliable query specifications.

Attributes of a type are always found by using paths in a ‘downward’ direction. It implies the use of an *its*-construct in the query language. This strict application of existing semantic attribute paths forbids the construction of arbitrary relationships through joins [1]. Variable or optional collections are found in the ‘upward’ direction. This implies the use of one of the set functions *max*, *min*, *total*, *count*, *any*, *nil*, and *some* together with the *per*-construct in the definition of a derived attribute. The above restrictions result in an orthogonal query language. Query formulations are unique without alternatives, which prevents pitfalls in user queries.

3 Applications

Several queries can be presented to illustrate recursive applications on our database. This section contains only a few of them to illustrate the concepts for data manipulation in this context. All queries illustrate that the facility to define derived attributes is crucial for problem solving in the Xplain environment. Each query starts with a title and short description. Then the formal query specification is followed by a brief explanation. Recursive queries contain always two elements: initialization (the starting condition) and recursion (the cascade update statement).

In complex query specifications the following two semantic statements are crucial: *extend* and *cascade*. Other statements in the query language are self-explanatory. For derivations the *extend* statement is globally defined as follows:

```
extend <subtype> its <extend attribute> =
        <attribute expression>³
        <function><maintype> its <attribute expression> <predicate>
                per <grouping attribute>
```

For recursive problems the *cascade* statement is globally defined as follows:

```
cascade <subtype> its <cascade attribute> =
        <attribute expression> <predicate> ³
        <function> <maintype> its <attribute expression> <predicate>
                per <grouping attribute>
```

Both statements have similar syntax. The differences between them are their semantics: *extend* is used to define a new <extend attribute> using the <attribute expression> or the <function> applied on <attribute expression> in groups of instances satisfying <predicate> of <maintype> according <grouping attribute> related to a <subtype> instance.

The *cascade* is used similarly as a semantic update driven by the <attribute expression> in instances satisfying the <predicate> or the <function> applied on <attribute expression> in groups of instances satisfying <predicate> of <maintype>

according <grouping attribute> related to a <subtype> instance. The required ordering can be determined from the query specification during query parsing using Lex and Yacc. The user does not need to specify the starting point(s) of recursive operations.

Following queries will illustrate application of these statements.

***Query 1*:** Determine the products containing a certain critical product.
This recursive query could be relevant when delivery or production of a certain critical product is delayed. It could play a role in determining the assemblies or finished products that are affected by a delay of this product. The user specifies the critical product in question.

value source = *input*(a25) "Enter product name: ". (1.1)
extend product *with* dependent = (*false*). (1.2)
update product *its* dependent = (*true*) *where* name = source. (1.3)
cascade product *its* dependent = (1.4)
  any fraction *where* minor_product *its* dependent (1.5)
  per major_product. (1.6)
get product *its* name (1.7)
  where name = source. (1.8)
get product *its* name *where* dependent *and* name ≠ source. (1.9)

Brief explanation:

(1.1) Alphanumeric input of at most 25 characters (a25) is expected from the user. The prompt consists of the message: ‘Enter product name: ’.

(1.2) All instances will start with ‘dependent’ extension initialized with the logical value *false*.

(1.3) Instances satisfying the input name are starting point for the cascade update statement. The starting point can consist of more than one product in case several products have the same name or in case a wildcard has been used in (1.1).

(1.4) Next the recurrent term can be specified. The attributes major_product (from target) and minor_product (from source) of fraction determine the processing order of the semantic cascade update statement. A minor_product must be processed before a major_product: processing can start with the basic products. Xplain DBMS determines this ordering during parsing of the cascade statement. In our case this ordering is determined with the assistance of Lex and Yacc. Note that the cascade statement cannot be replaced by an extend statement because it is impossible to define a derived attribute using some operation on itself.

(1.7) The result consists of two lists. The first list is the input product(s).

(1.9) The second list of the result consists of dependent products (assemblies or finished products), not belonging to the first list that contains the product in question.

A similar query can be expressed using the reverse ordering (e.g. Determine the components of a certain product). The symmetry of structuring concepts in the data model and the symmetry of the manipulation concepts in the query language are then demonstrated (cf. query 2).

***Query 2*:** Create production/purchase order information.
Ordered quantities of finished products or intermediate assemblies have consequences for orders of basic components, taken into account the stock of its components. In this recursive query two lists are produced. One list gives the ordered products; the other gives the needed basic components and the quantities needed to produce the ordered products. Products may occur in several assemblies and/or finished products; the structure does constitute a hierarchy (i.e. a directed acyclic graph) and not necessarily a tree (see figure 4).

get product *where* ordered_quantity > 0. (2.1)

extend product *with* need = ordered_quantity - stock. (2.2)

extend product *with* induced = 0. (2.3)

cascade product *its* induced = (2.4)

total fraction *its* multiplicity * (major_product *its* need + major_product *its* induced)

per minor_product. (2.6)

extend product *with* basic = *nil* fraction *per* major_product. (2.7)

update product *its* need = need + induced. (2.8)

get product *its* name, need *where* basic *and* need > 0. (2.9)

Brief explanation:

(2.1) First a list of ordered products is created and presented to the user.

(2.2) For each product the number of additional needed products can be calculated.

(2.3) Nothing is beforehand assumed about the logical ordering of products, so initially products induce nothing from higher assemblies. Xplain determines the ordering using Lex and Yacc.

(2.4) The totality of products needed of a minor_product is determined by the given calculation of the attribute expression. In the calculation the number of products needed as well as the number of products induced by its higher assemblies must be taken into account.

(2.6) This sum (major_product *its* need + major_product *its* induced) cannot be calculated using the extend statement because the number of iterations is not known beforehand.

(2.7) Basic products must be determined.

(2.8) The total quantity needed, consists of its own quantity needed and the quantity needed on the basis of the fractions from higher assemblies.

(2.9) For each needed basic product are given: the identification (is always given), its product name and the quantity that must be ordered or produced.

Query 3**:** Determine for each product the level of compound.
In this recursive query the topological ordering of products is determined. For each product is needed: identification, product name, level of compound. A useful application of this query is found in just-in-time production lines where each production level requires a certain production time. The result of this query can be used to determine critical delivery dates of basic components.

extend product *with* level = 1. (3.1)
cascade product *its* level = (3.2)
  max fraction *its* minor_product *its* level + 1 (3.3)
  per major_product. (3.4)
value levels = *max* product *its* level. (3.5)
value text = *combine*(" Number of levels: ", levels). *value* text. (3.6)
get product *its* name, level *per* -level. (3.7)

Brief explanation:
(3.1) The fact that level occurs in the result with numeric domain can be determined during parsing using Lex and Yacc.
(3.3) In the previous query we used the *total* function. This query is an illustration of an application with the *max* function on collections.
(3.5) The maximum level is determined. Its definition is called 'levels'.
(3.6) The result of this query should also include the value of 'levels'. This value is presented in a string consisting of the substring 'Number of levels: ' and the number. Function *combine* is a string concatenation function. The constructed string is displayed by using the second *value* statement.
(3.7) The list is ordered by descending level (so finished products are given first). Without '*per* -level', products are presented in a system defined order.

4 Performance

A product structure consists normally of hundreds (e.g. electric shaver), thousands (e.g. television) or much more (e.g. production robot) assemblies/basic products. Characteristic for all these hierarchies is that they consist of only a few levels (e.g. 2 to 10) with considerably more basic products than finished products. Figure 4 gives a simplified example in which a node represents a product and an arc represents a fraction.

It is evident that any extend or get statement is of linear processing order. For the cascade statement first an ordering of fractions is required. This ordering is determined by the structure of the hierarchy and the query specification as well i.e. the *per*-construct. As the term *cascade* indicates, the ordering of fractions is a repeated version of a not-recursive query.

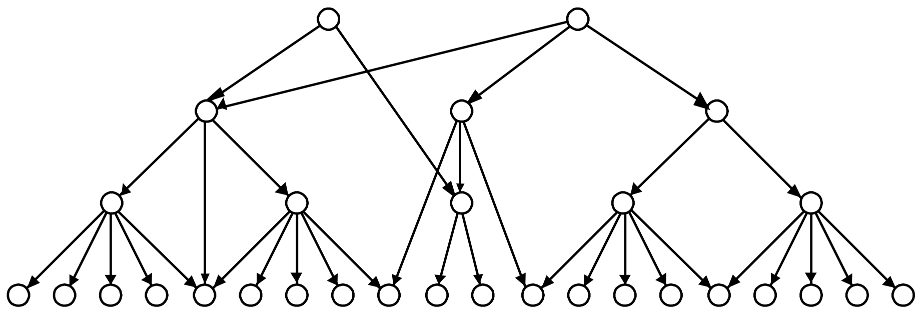


Fig. 4. Example of a product structure

An iteration process in which the relevant extension is repeatedly executed determines the ordering of fractions. Each iteration step results in some referring fractions to be processed in the cascade statement. This process reduces the number of relevant products and fractions until no more fractions are left for processing or a cycle in the structure is detected. The processing time is therefore utmost proportional to the product of the number of levels (which is limited to a small number) and the number of fractions. This implies that the processing complexity should be linear in a practical environment, which also applies to the detection of cycles. Details about the algorithm will be published in a separate paper.

In case of cycles, query processing is aborted and an appropriate message (including a list of instances causing the cycle) is given. This listing is evident because generation of all individual cycles may lead to an unsolvable problem. The sketched approach implies that the software system can take its responsibility: cycles in a directed graph are detected by the system in linear time, which prevents infinite processing.

Simple test cases have confirmed this outcome for the product database applications. Three databases of different sizes have been generated. Table 2 gives an overview of execution times in seconds of some not-recursive queries together with the purchase order application (Query 2) using a 333 MHz Pentium II notebook under standard Linux. In this table the term *outdegree* refers to the average number of fractions per assembly.

Table 2. Performance on different databases

Database Size: (levels, outdegree, products)	small (3, 5, 259)		medium (5, 5, 1555)		large (6, 5, 9330)	
Query:	result	time	result	time	result	time
get fractions	# 252	0.02	# 1551	0.12	# 9327	0.83
determine finished products	# 7	0.01	# 6	0.03	# 6	0.16
determine basis products	# 217	0.02	# 1297	0.11	# 7776	0.60
count product and fraction	# 2	0.01	# 2	0.01	# 2	0.05
purchase order (Query 2)	# 188	0.08	# 647	0.47	# 5191	3.58

The total elapsed time for foregoing queries includes: parsing, total execution time, preparation of the result (see #-value above) and presentation of the first 20 instances on screen. The function clock () from the standard C-library has been used for timing.

Elapsed times for the purchase order query are influenced by the preparation of the results (ranging from 188 to 5191 instances). Clearance of the preparation time from the elapsed time results in a linear execution time with a factor ranging from 0.25 to 0.34 sec. per 1000 fractions on the notebook.

The ordering algorithm does not possess a special position within Xplain DBMS: its implementation makes also use of files in the existing infrastructure of the Xplain file system. It is, like all other data handling, not implemented as operating on main memory data but on disk-stored data. Considerable improvements can therefore be expected when the algorithm is further tailored for speed.

5 Discussion

It is clear that the main contributions of this paper do not lie in the introduction of new concepts for the solution of recursive database problems. On the contrary its main contributions are based on the fact that practical recursive problems can easily be solved by using existing standard semantic concepts when the relevant recursive processing aspects are delegated completely to the software system and are not to the user. That is also the reason why complete language syntax is missing; this syntax can already be found in [9].

It is extremely important that designer and end user for recursion completely rely on the intelligence and efficiency of the semantic software system. This is new and only a mental shift made the developments possible. Unawareness about this crucial separation between declarative specification and query processing has caused a major delay in our own research developments.

Afterwards the result is obvious. Relational query languages make it already possible to specify queries without explicit iteration because iteration aspects do not play any role in the relational model. Therefore the user can ignore iteration. Because iteration is system's responsibility nothing can go wrong with that (for example users cannot specify infinite processes). So afterwards it is evident that users can only neglect recursion if recursion does not play any role in data model definitions.

Although processing details are hidden in the query specification, the user can completely ignore them. Also here the same benefits can be expected: nothing can go wrong with recursive applications. Processing recursive queries is only system's responsibility and the system has the obligation to generate appropriate messages (in reasonable time of course) in case recursion cannot be performed.

Nested data models are incompatible with our goal because they require explicit navigation through data, which is conflicting with the desired separation between declarative specification and query processing.

Conclusion

The advantages of semantic data modeling for recursive applications have been demonstrated. Because query specification is associative, navigation and instance processing plays no role in the specification. This has enormous consequences for user's performance and system's performance of query language solutions. Navigation from one instance to another (as done in programming) has been replaced by an associative approach in which only structural relationships play a role. End users are using only one paradigm and do not have to worry about the order in which data in a database are processed and finiteness of processes is in all cases guaranteed by the DBMS. Semantic information gives the DBMS also the opportunity to find the most optimal processing strategy. Performance tests have confirmed that product database applications can be processed in linear time.

References

1 J.A. Bakker and J.H. ter Bekke, Foolproof Query Access to Search Engines, Proc. 3rd Int. Conf. on Information Integration and Web-based Applications & Services (IIWAS 2001), Linz, Austria (2001), 389-394.

2 J.A. Bouma and J.H. ter Bekke, Getting Information from Dispersed Databases through Hyperqueries, Proc. Int. Conf. on Applied Informatics (AI 2002), Innsbruck, Austria (2002), 443-447.

3 M. Davis, Unsolvable Problems, In: Handbook of Mathematical Logic, J. Barwise (ed.), North-Holland, Amsterdam (1977), 568-593.

4 M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman, New York (1979).

5 D.M. Kroenke, Database Processing: Fundamentals, Design, and Implementation (8th edition), Prentice Hall, Upper Sadle River, NJ (2002).

6 F. Rolland, The Essence of Databases, Prentice Hall, Hemel Hemstead (1998).

7 P. Schauble and B. Wuthrich, On the Expressive Power of Query Languages, ACM Trans. on Information Systems, Vol 12., No. 1, January 1994, 69-91.

8 D. Suciu, J. Paredaens, Any Algorithm in the Complex Object Algebra with Powerset Needs Exponential Space to Compute Transitive Closure, Proc. 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (1994), page 201.

9 J.H. ter Bekke, Semantic Data Modeling, Prentice Hall, Hemel Hempstead (1992).

10 J.H. ter Bekke, Advantages of a Compact Semantic Meta Model, Proc. 2nd IEEE Metadata Conference, Silver Spring (1997).

11 J.H. ter Bekke, Can We Rely on SQL?, Proc. DEXA Workshop, Toulouse (1997), IEEE Computer Society, 378-383.

12 J.H. ter Bekke and J.A. Bakker, Limitations of Relationships from Coinciding Data, Proc. Int. Conf. Intelligent Systems and Control (ICSC 2001), Clearwater, Florida (2001), 247-252.

13 J.H. ter Bekke and J.A. Bakker, Content-driven Specifications for Recursive Project Planning Applications, Proc. Int. Conf. on Applied Informatics (AI 2002), Innsbruck, Austria (2002), 448-452.

A Class-Based Logic Language for Ontologies

Djamal Benslimane[1], Mohand-Saïd Hacid[1], Evimaria Terzi[2], and Farouk Toumani[3]

[1] LISI - UFR Informatique
Université Claude Bernard Lyon 1
8, Blvd Niels Bohr
69622 Villeurbanne - France
dbenslim@iuta.univ-lyon1.fr, mshacid@lisi.univ-lyon1.fr
[2] Department of Computer Sciences, Purdue University
West Lafayette, IN 47907, USA
edt@cs.purdue.edu
[3] LIMOS-ISIMA
Campus des Cézeaux - B.P. 125
63173 AUBIERE - France
ftoumani@sp.isima.fr

Abstract. We describe a formal design of a hybrid language and show that it provides a suitable basis for representing and interacting with ontologies. We specify ontologies and query them with a clear logical semantics. The proposed language combines the expressive power of description logics and rules.

Keywords: Ontologies, Description Logics, Rule-Based Languages.

1 Introduction

In the last decade there has been a growing recognition of the need to create explicit specifications of how knowledge in a domain is conceptualized. Such an explicit specification is now normally termed an "ontology" in the literature [6]. An ontology is considered as a set of logical axioms designed to account for the intended meaning of a vocabulary [12]. Ontologies are very useful for interoperability and for browsing and searching. The importance of ontologies is being recognized in diverse research fields such as knowledge representation, knowledge engineering, database design, information integration, and information systems.

Interest in ontologies has grown as researchers and system developers have become more interested in reusing or sharing knowledge across systems. Currently, one key impediment to sharing knowledge is that different systems use different concepts and terms for describing domains. These differences make it difficult to take knowledge out of one system and use it in another. If we could develop ontologies that could be used as the basis for multiple systems, they would share a common terminology that would facilitate sharing and reuse. Developing such reusable ontologies has been an important goal of ontology research. Similarly, if we could develop tools that would support merging ontologies and translating between them, sharing would be possible even between systems based on different ontologies.

In case of a large network of autonomous and heterogeneous databases as those potentially accessible from the Web, a meaningful organization and segmentation of databases would have to be based on simple *ontologies* that describe coherent slices of the information space. These distributed ontologies would filter interactions, accelerate information searches, and allow for the sharing of data in a tractable manner.

In this paper we follow this approach. We consider the semantic of each source data/knowledge base described by means of a formal ontology. Then we provide a *formal specification language* and a *constraint query language* that can be used to *infer* relationships about ontologies. Reasoning about ontologies is not only relevant for optimization or filtering, but it can also be applied to organize large sets of ontologies into groups which can be important to support *navigation.*

Paper outline: Section 2 summarizes the contributions of this paper. Section 3 introduces preliminary notions. Section 4 formally defines our specification language. In Section 5 we develop our query language. The query language can be seen as consisting of a constraint language on top of which relations can be defined by definite clauses. Section 6 surveys related work. We conclude in Section 7, with a recapitulation of our work and a brief overview of the perspectives it offers.

2 Contributions

Maybe not so obvious, but nevertheless very important, is the use of ontologies in a connection with the user interface component [12]. At run time, the first role ontologies can play within the user interface is to allow themselves to be *queried* and *browsed* by the user. In this case, the user is aware of the ontologies, and uses them as part of his normal use of the information system. However, until now, the provided interfaces (i.e., query languages) and reasoning facilities (see Section 6) are rather limited. Therefore, there is a need for formal tools that organize ontologies so as to provide effective access and retrieval.

The usefulness of organizing ontologies in a hierarchical fashion has been amply recognized. Our work is also a contribution to the subject from the point of view of *querying terminological ontologies.*

To put the whole subject on a firm logical basis, we rely on the ideas of CLP[1] schemes. In this view, queries and answers are both seen as constraints,

[1] Constraint Logic Programming.

the latter in some "canonical" or "solved" form intended to capture the idea of a useful answer. The knowledge base which defines a terminological ontology comprises both a concept hierarchy, and a constraint logic program that defines relations among the concepts in the form of constraints. The query language is richer than the language in which the knowledge pertaining to the ontologies can be expressed, and is essentially a variant of first-order logic with terms denoting concepts (i.e., types, constraints) or roles, instead of individuals.

We describe a logical (clausal) query language that can be used to interact with ontologies described in the description logic, and infer relationships between ontologies. We view our query language as consisting of a constraint language on top of which relations can be defined by definite clauses. The language has a higher-order syntax to allow ontology browsing, and a declarative and operational semantics. We show how *resolution* and *classification* can be used together within a common framework to interact with complex ontologies organized in concept hierarchies. To manipulate ontologies, our language has an interpreted function symbol for building new ontologies from old ones. Intuitively, if c_1 and c_2 are two concepts defined in the ontologies X and Y respectively, then the term $c_1 \uplus c_2$ denotes the new ontology containing c_1 and c_2 and all their subconcepts. The semantics is based on fixpoint theory, as in classical logic programming, and on a new notion of active domain, called the *extended active domain*, which is introduced to allow for a *bottom-up evaluation* of rules.

A knowledge base describing an ontology contains both a set of constrained rules and a description-logic terminology. We combine the two formalisms by allowing concepts and roles, defined in the terminology, to appear as arguments of predicates in the rules. The semantics of the resulting language is derived naturally from the semantics of its component languages.

3 Preliminaries

To make the paper self-containing, we shall introduce some notions that are necessary for our application.

3.1 Description Logics

Description logics (also called concept logics, terminological logics, or concept languages) are a family of logics designed to represent the taxonomic and conceptual knowledge of a particular application domain on an abstract, logical level. They are equipped with well-defined, set-theoretic semantics. Furthermore, the interesting reasoning problems such as subsumption and satisfiability are, for most description logics, decidable (see, for example, [8]).

Starting from atomic concepts and roles[2], complex concepts (and roles) are built by using a set of constructors. For example, from atomic concepts

[2] A concept is interpreted as a class of objects in the domain of interest, and then can be considered as an unary predicate. Roles are interpreted as binary relations on individuals, and then considered as binary predicates.

`Human` and `Female` and the atomic role `child` we can build the expression $\texttt{Human} \sqcap \forall\texttt{child.Female}$ which denotes the set of all `Human` whose children are (all) instances of `Female`. Here, the symbol $\sqcap$ denotes conjunction of concepts, while $\forall$ denotes (universal) value restriction.

A knowledge base in a description logic system is made up of two components: (1) the *TBox* is a general schema concerning the classes of individuals to be represented, their general properties and mutual relationships; (2) the *ABox* contains a partial description of a particular situation, possibly using the concepts defined in the *TBox*. It contains descriptions of (some) individuals of the situation, their properties and their interrelationships.

Retrieving information in a knowledge base system based on description logics is a deductive process involving both the schema (*TBox*) and its instantiation (*ABox*). In fact, the *TBox* is not just a set of constraints on possible *ABoxes*, but contains intensional information about classes. This information is taken into account when answering queries to the knowledge base. The following reasoning services are the most important ones provided by knowledge representation systems based on description logics (See [10] for an overview):

– *Concept satisfiability*: Given a knowledge base and a concept C, does there exist at least one model of the knowledge base assigning a non-empty extension to C?
– *Subsumption*: Given a knowledge base and two concepts C and D, is C more general than D? That is, is each instance of D also an instance of C in all models of the knowledge base?
– *Knowledge base satisfiability*: Are an *ABox* and a *TBox* consistent with each other? That is, does the knowledge base admit a model?
– *Instance checking*: Given a knowledge base, an individual o, its (partial) description, and a concept C, is o an instance of C in any model of the knowledge base?

3.2 Constraints

A *constraint* is some piece of syntax constraining the values of the variables occurring in it. It is a restriction on a space of possibilities. Mathematical constraints are precisely specifiable relations among several unknowns (or variables), each taking a value in a given domain. Constraints restrict the possible values that variables can take, representing some (partial) information about the variables of interest. Constraints enjoy several interesting properties. First, as said before, constraints may specify partial information –a constraint need not uniquely specify the value of its variables. Second, they are additive: given a constraint c_1 say, $X + Y \geq Z$, another constraint c_2 can be added, say, $X + Y \leq Z$. The order of imposition of constraints does not matter; all that matters at the end is that the conjunction of constraints is in effect. Third, constraints are rarely independent; for instance, once c_1 and c_2 are imposed it is the case that the constraint $X + Y = Z$ is entailed. Fourth, they are nondirectional: typically a constraint on (say) three variables X, Y, Z can be used to infer a constraint on X given constraints on Y and Z, or a constraint on Y given constraints on X and Z and so on. Fifth, they are declarative: typically they specify what relationship must hold without specifying a computational procedure to enforce that relationship. Any computational system dealing with constraints must fundamentally take these properties into account.

4 Modeling Ontologies : A Description Logic Approach

Ontologies can be classified according to two dimensions [24]: *the amount and type of structure of the conceptualization* and *the subject of the conceptualization*. With respect to the first dimension, three categories can be distinguished:

1) *Terminological Ontologies* such as lexicons, specify the terms that are used to represent knowledge in the domain of discourse. An example of such an ontology in the medical field is the semantic network in UMLS (Unified Medical Language System) [16].
2) *Information Ontologies* which specify the record structure of databases. Database schemata are an example of this class of ontologies. Level 1 of the PEN and PAD model [22], a framework for modeling medical records of patients, is a typical example of such an ontology in the medical field. At this level, the model provides a framework for recording the basic observations of patients, but it makes no distinction between symptoms, signs, treatments, etc.
3) *Knowledge Modeling Ontologies* specify conceptualizations of the knowledge. Compared to information ontologies, knowledge modeling ontologies usually have a richer internal structure. Furthermore, these ontologies are often tuned to a particular use of the knowledge that they describe. Within the context of our project, knowledge modeling ontologies are the ontologies that we are mostly interested in.

We first introduce the syntax and semantics of a description logic for representing ontologies as well as the subsumption problem. *Concept descriptions* are inductively defined by means of a set $\mathcal{C}$ of *primitive concepts* (unary predicates), a set $\mathcal{R}$ of *role names* (binary predicates), and a set of *constructors*. The semantics of a concept description is also inductively defined whereby primitive concepts are interpreted as subsets of a domain Δ and role names are interpreted as binary relations on $\Delta \times \Delta$.

Definition 1. (Syntax) *Let $\mathcal{C}$ be a set of concept names and $\mathcal{R}$ be a set of role names. Let $R \in \mathcal{R}$ be a role name and $n \in \mathbb{N}$. Concepts C and D can be formed by means of the following syntax:*

$$\begin{array}{lll}
C, D \longrightarrow & A \mid & \text{(atomic concept)} \\
& \top \mid \bot \mid & \text{(top, bottom)} \\
& C \sqcap D \mid C \sqcup D \mid & \text{(conjunction, disjunction)} \\
& \neg C \mid & \text{(complement)} \\
& \forall R.C \mid & \text{(value restriction)} \\
& \exists R.C \mid & \text{(existential quantification)} \\
& (\leq nR) \mid & \text{(at most number restriction)} \\
& (\geq nR) & \text{(at least number restriction)}
\end{array}$$

Definition 2. (Semantics) *The semantics is given by an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ which consists of an (abstract) interpretation domain $\Delta^{\mathcal{I}}$, and an interpretation function $\cdot^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ maps each atomic concept $A \in \mathcal{C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta$ and each role name $R \in \mathcal{R}$ to a subset $R^{\mathcal{I}} \subseteq \Delta \times \Delta$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concepts is inductively defined as follows:*

$$\begin{aligned}
A^{\mathcal{I}} &= A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \exists y : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \forall y : (x, y) \in R^{\mathcal{I}} \longrightarrow y \in C^{\mathcal{I}}\} \\
(\leq nR)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \quad \|\{y \mid (x, y) \in R^{\mathcal{I}}\}\| \leq n\} \\
(\geq nR)^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \quad \|\{y \mid (x, y) \in R^{\mathcal{I}}\}\| \geq n\}
\end{aligned}$$

In the following, for the sake of simplicity, we rewrite $(\forall R.C)$ as (all R C), $(\leq nR)$ as (atmost n R), and $(\geq nR)$ as (atleast n R).

Let A be a concept name and let D be a concept term. Then $A = D$ is a terminological axiom (also called definition). A terminology (*TBox*) is a finite set $\mathcal{T}$ of terminological axioms with the additional restriction that no concept name appears more than once as a left hand side of a definition.

An interpretation $\mathcal{I}$ is a model for a *TBox* $\mathcal{T}$ if and only if $\mathcal{I}$ satisfies all the assertions in $\mathcal{T}$.

In this paper, we consider that ontologies are specified by means of a set of terminological axioms. Hence, each ontology is a terminology.

One of the most important inference tasks we are interested in is computing the *subsumption relation* between concepts, i.e., deciding the question if one concept is more specific than another.

Definition 3. (Subsumption) *Let C, D be two concepts. D subsumes C (for short $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$. C is equivalent to D (for short $C \equiv D$) iff $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e., $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.*

In the literature several approaches to subsumption have been investigated (see [10] for an overview). In order to decide subsumption for very expressive languages we can employ tableaux-based algorithms (see, e.g., [23,5,3]). The automata theoretic approach has been proposed in order to gain a more profound understanding of the semantics as well as the subsumption relation in cyclic terminologies for rather small languages [21,2,14]. On the other hand, structural subsumption [4,7] algorithms are efficient methods for deciding subsumption of concept descriptions that do not use full negation, disjunction or existential restrictions.

The description logic $\mathcal{ALCNR}$ [5] allows for more constructors than our language, i.e., role conjunction. Since we are concerned with a sublanguage of $\mathcal{ALCNR}$, and as subsumption for $\mathcal{ALCNR}$ is *decidable* [5], so it is for the language proposed here.

In the rest of the paper, we use equally the terms ontology, knowledge base, and terminology.

4.1 Example

Consider the database recording information about transportation. This database is augmented with an ontology. The ontology is specified as a set of terminological axioms. Figure 1 shows a small fragment of such an ontology.

Product = (atmost 1 *price*) ⊓ (atleast 1 *price*) ⊓ (all *price* **Number**)
Trip = **Product** ⊓ (atmost 1 *departure_time*) ⊓ (atleast 1 *departure_time*) ⊓ (all *departure_time* **Time**)⊓
(atmost 1 *arrival_time*) ⊓ (atleast 1 *arrival_time*) ⊓ (all *arrival_time* **Time**)⊓
(atmost 1 *departure_date*) ⊓ (atleast 1 *departure_date*) ⊓ (all *departure_date* **Date**)⊓
(atmost 1 *arrival_date*) ⊓ (atleast 1 *arrival_date*) ⊓ (all *arrival_date* **Date**)⊓
(atmost 1 *departure_country*) ⊓ (atleast 1 *departure_country*) ⊓ (all *departure_country* **String**)⊓
(atmost 1 *arrival_country*) ⊓ (atleast 1 *arrival_country*) ⊓ (all *arrival_country* **String**)⊓
(atmost 1 *initial_rate*) ⊓ (atleast 1 *initial_rate*) ⊓ (all *initial_rate* **Number**)⊓
(atmost 1 *departure_place*) ⊓ (atleast 1 *departure_place*) ⊓ (all *departure_place* **Place**)⊓
(atmost 1 *arrival_place*) ⊓ (atleast 1 *arrival_place*) ⊓ (all *arrival_place* **Place**)⊓
(atmost 1 *transport_mean*) ⊓ (atleast 1 *transport_mean*) ⊓ (all *transport_mean* **String**)
TrainTrip = **Trip** ⊓ (atmost 1 *departure_period*) ⊓ (atleast 1 *departure_period*) ⊓ (all *departure_period* **String**)⊓
(atmost 1 *train_type*) ⊓ (atleast 1 *train_type*) ⊓ (all *train_type* **String**)⊓
(atmost 1 *discount*) ⊓ (atleast 1 *discount*) ⊓ (all *discount* **String**)
PlaneTrip = **Trip** ⊓ (atmost 1 *plane_type*) ⊓ (atleast 1 *plane_type*) ⊓ (all *plane_type* **String**)⊓
(atmost 1 *discount*) ⊓ (atleast 1 *discount*) ⊓ (all *discount* **String**)
Lodging = **Product** ⊓ (atmost 1 *number_of_nights*) ⊓ (atleast 1 *number_of_nights*)⊓
(all *number_of_nights* **Number**)⊓
(atmost 1 *initial_rate*) ⊓ (atleast 1 *initial_rate*) ⊓ (all *initial_rate* **Number**)⊓
(atmost 1 *location*) ⊓ (atleast 1 *location*) ⊓ (all *location* **Place**)
Accommodation = (atmost 1 *initial_rate*) ⊓ (atleast 1 *initial_rate*) ⊓ (all *initial_rate* **Number**)⊓
(atmost 1 *location*) ⊓ (atleast 1 *location*) ⊓ (all *location Place*)⊓
(atmost 1 *category*) ⊓ (atleast 1 *category*) ⊓ (all *category* **String**)
Hotel = **Accommodation** ⊓ (atmost 1 *number_of_beds*) ⊓ (atleast 1 *number_of_beds*)⊓
(all *number_of_beds* **Number**)⊓
(atmost 1 *included_service*) ⊓ (all *included_service* **String**)
House = **Accommodation** ⊓ (atmost 1 *designation*) ⊓ (atleast 1 *designation*) ⊓ (all *designation* **String**)⊓
(atmost 1 *number_of_rooms*) ⊓ (atleast 1 *number_of_rooms*) ⊓ (all *number_of_rooms* **Number**)
Apartment = **Accommodation** ⊓ (atmost 1 *designation*) ⊓ (atleast 1 *designation*) ⊓ (all *designation* **String**)⊓
(atmost 1 *number_of_rooms*) ⊓ (atleast 1 *number_of_rooms*) ⊓ (all *number_of_rooms* **Number**)
Camping = **Accommodation** ⊓ (atmost 1 *designation*) ⊓ (atleast 1 *designation*) ⊓ (all *designation* **String**)⊓
(atmost 1 *number_of_places*) ⊓ (atleast 1 *number_of_places*) ⊓ (all *number_of_places* **Number**)⊓
(atmost 1 *category*) ⊓ (atleast 1 *category* ⊓ (all *category* **String**)
. . .

Fig. 1. Ontology example specified by terminological axioms

In our case, ontologies, in the form of terminologies are named. Let us call this ontology Transportation. The first declaration (regarding Product) is intended to express necessary and sufficient conditions that must be met by each instance of the category Product. For every instance of product, the *price* attribute must relate to it exactly 1 filler, itself instance of the basic type Number. In this description, Number is a basic type, while *price* is the name of a binary relation, intended to relate products to numbers. For this kind of statement, the left hand side of the symbol = is called concept name, and right hand side is called the definition of the concept (or concept term). There are several things one can do with such a description. The most important one is *reasoning* about the relationship of one description to another, treating them as "intentional" objects. For example the description of Product *subsumes* (is *entailed by*) the description of TrainTrip.

From the definitions of figure 1, the hierarchy of figure 2 can be generated automatically thanks to the subsumption relation. The nodes in the graph represent the definitions (i.e., ontological theories), and the edges denote inclusion relations. This ontology is not based on a fixed hierarchy, but on a framework of distinctions, from which the hierarchy is generated automatically. When application-dependent distinctions are added to the basic set, a new hierarchy is created.

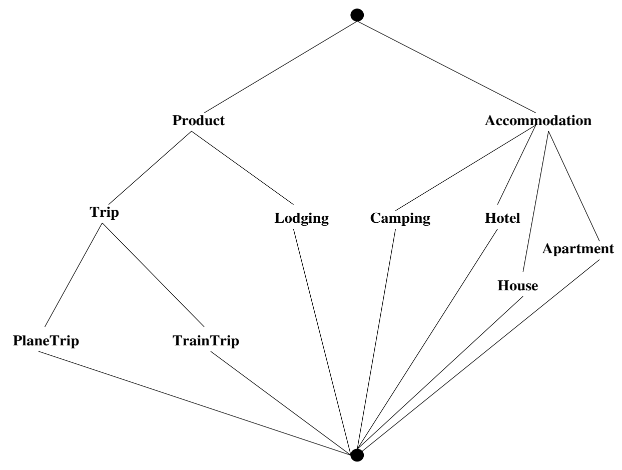


Fig. 2. A concept hierarchy corresponding to the ontology of figure 1

Let c be a concept name and k be a knowledge base. We denote by def_{c_k} the definition (if any) associated with the concept c in the knowledge base k.

Let k_1 and k_2 be two knowledge bases. k_1 is a *refinement* of k_2 iff $\forall c \in k_1, \exists c' \in k_2 : def_{c_{k_1}} \sqsubseteq def_{c'_{k_2}}$

if two ontologies are *refinements* of each other, then they must be *isomorphic*.

5 A Language for Querying Ontologies

Rule-based languages lie at the core of several areas of central importance in databases and artificial intelligence, such as deductive databases and production systems.

In this section, we present a declarative, rule-based query language that can be used to *browse* ontologies. The language is based on a CLP scheme. It has a model-theoretic and fixpoint semantics based on the notion of *extended active domain* of a database. The language has an interpreted function symbol for building new knowledge bases (i.e., ontologies) from others (by importing concepts from them). A constructive term has the form $A \uplus B$ and it is interpreted as the union of the two concepts A and B, together with all their subconcepts.

A knowledge base in our language contains both a set of constrained rules and a description-logic terminology. We combine the two formalisms by allowing concept names and role names to appear as arguments of predicates. Closest to our work are the investigations in [15,9,17]. However, their frameworks and results are different.

5.1 Syntax

The language of terms uses four countable, disjoint sets:

1. A set $\mathcal{D}$ of constant symbols. This set is the union of three disjoint sets:
   - $\mathcal{D}_1$: a set of concept names,
   - $\mathcal{D}_2$: a set of role names,
   - $\mathcal{D}_3$: a set of knowledge base (i.e., ontology) names.
2. A set $\mathcal{V}$ of variables called concept variables and denoted by $X, Y, \ldots$
3. A set $\tilde{V}$ of variables called role variables and denoted by $x, y, \ldots$
4. A set $\bar{V}$ of variable called knowledge base variables and denoted by $\alpha, \beta, \ldots$

If T_1 and T_2 denote concept names, concept variables or constructive ontology terms, then $T_1 \uplus T_2$ is a *constructive* ontology term. The merging operator $\uplus$ is such that: $\forall c_1, c_2 \in \mathcal{D}_1 \; c_1 \uplus c_2$ is a new knowledge base.

Let $c_1 \in k_1$ and $c_2 \in k_2$, where c_1 and c_2 are two concept names, and k_1 and k_2 are two knowledge base names. The structure of the knowledge base $k = c_1 \uplus c_2$ resulting from importing c_1 and c_2 is such that:

1. $id_k = f(id_{c_1}, id_{c_2})$. Here, we follow the idea of [13] that the object *id* of the knowledge base generated from c_1 and c_2 should be a function of id_{c_1} and id_{c_2}.
2. k is an abbreviation of $c_1.c_2$; that is, the name of the newly generated knowledge base is the concatenation of the two concept names involved in the function symbol $\uplus$.
3. $c_1 \in k$ and $c_2 \in k$
4. $\forall c : c \sqsubseteq c_1$ in $k_1 \Rightarrow c \sqsubseteq c_1$ in k
5. $\forall c' : c' \sqsubseteq c_2$ in $k_2 \Rightarrow c' \sqsubseteq c_2$ in k

For example, given the ontology hierarchy of figure 2, $\mathsf{Trip} \uplus \mathsf{Camping}$ leads to a subset of the terminology of figure 1 whose subsumption hierarchy is shown in figure 3.

Note that $T_1 \uplus T_1 = T_1$. This leads to the termination of the execution of constructive rules (i.e., rules containing a constructive term in their heads).

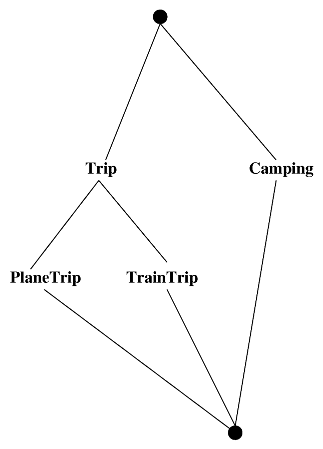


Fig. 3. Hierarchy derived from figure 2 by performing the operation Trip $\uplus$ Camping

Definition 4. (Predicate Symbols). *We define the following predicate symbols:*

- *A special unary predicate symbol* **ontology***. It can be seen as the class of all ontologies. Each ontology is seen as a terminology identified with a name. For the example of Subsection 4.1 (figure 1), we have the fact* **ontology**$(Transportation)$.
- *A special binary predicate symbol* **concept***.* $concept(a, b)$ *means the knowledge base a contains the defined concept b. Here, a and b stand for a knowledge base name and a concept name respectively. For example,* **concept**$(Transportation,\ Trip)$, **concept**$(Transportation,\ Product)$ *hold for the example of figure 1.*
- *A special binary predicate symbol* **role***.* $role(a, r)$ *means the role r is used in the terminology a. Examples are* **role**$(Transportation, price)$ *and* **role**$(Transportation, designation)$ *in the case of figure 1.*
- *The other predicates are ordinary predicates.*

Hence, the only extensional relations are **ontology**, **concept**, and **role**.

Definition 5. (Atom). *If P is an n-ary predicate symbol and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is an atom.*

Definition 6. (Rule). *A rule in our language has the form:*

$$r : H \leftarrow L_1, \ldots, L_n \| c_1, \ldots, c_m$$

where H is an atom, $n, m \geq 0$, $L_1, \ldots, L_n$ are positive literals, and $c_1, \ldots, c_m$ are terminological constraints.

Optionally, a rule can be named as above, using the prefix "r :", where r is a constant symbol. We refer to H as the head of the rule and refer to $L_1, \ldots, L_n \| c_1, \ldots, c_m$ as the body of the rule.

Note that we impose the restriction that constructive terms appear only in the head of a rule and not in the body. A rule that contains a constructive term in its head is called a constructive rule.

Regarding the evaluation of the rule, $L_1, \ldots, L_n$ will be tested for satisfiability by the assertional component, while $c_1, \ldots, c_m$ will be tested for satisfiability by the terminological component.

Definition 7. (Range-restricted Rule). *A rule r is said to be range-restricted if every variable in the rule occurs in a body literal. Thus every variable occurring in the head occurs in a body literal.*

Definition 8. (Program). *A program is a collection of range-restricted rules.*

Definition 9. (Query). *A query is of the form:*

$$Q :?q(\bar{s})$$

where q is referred to as the query predicate and $\bar{s}$ is a tuple of constants and variables.

Examples. Let us illustrate our syntax by some query examples:

1. The following query will return all the concepts which are subconcepts of the concept *People* in any ontology.

$$\mathsf{answer}(X) \leftarrow \mathsf{ontology}(\alpha), \mathsf{concept}(\alpha, X) \| X \sqsubseteq People$$

   The constraint $X \sqsubseteq People$ (which is a subsumption constraint) will be tested for satisfiability by the terminological component of the system.
2. The query

$$\begin{aligned} \mathsf{answer}(X, y) \leftarrow\ & \mathsf{ontology}(State\_patrol), \\ & \mathsf{concept}(State\_patrol, X), \\ & \mathsf{role}(State\_patrol, y) \| X \sqsubseteq (\mathsf{atleast}\ 3\ y) \end{aligned}$$

   will return a set of pairs of concepts and roles, such that the concept X is defined in the ontology *State_patrol* and there is a constraint $(\mathsf{atleast}\ 3\ y)$ attached to the role y in the definition of X. This is equivalent to say that X is subsumed by $(\mathsf{atleast}\ 3\ y)$.

3. Suppose we have a set of ontologies. The following rule takes two ontologies and builds a new ontology whose terminological axioms are X and subconcepts of X, in each original ontology.

   $\mathsf{build\_new\_ontology}(X \uplus X) \leftarrow \mathsf{ontology}(\alpha),$
   $\quad \mathsf{ontology}(\beta), \mathsf{concept}(\alpha, X), \mathsf{concept}(\beta, X)$

   This is an example of a query allowing to find commonalities between two different ontologies and deriving a new ontology. The new ontology may be used as an intermediary between the systems based on the original ontologies.

   The simplest semantics of inheritance accounting for subtyping can be expressed as set intersection. A concept is subsumed by another concept if it denotes a subset of the other concept's denotation. This is not only natural, but also effective in expressing the semantics of multiple inheritance as set intersection. Thus is inheritance used in the concept languages. In fact set inclusion is used to characterize approximations of objects, and intersection as a means to combine object attributes. In our case, we use union, which is more natural in the framework on ontologies for databases, to combine object attributes. For example -using the informal notation $X : C$ to mean "object X is an instance of the concept C"- if $X : (\mathsf{all}\ R\ C)$ and $X : (\mathsf{all}\ R\ C')$, then $X : (\mathsf{all}\ R\ (C \sqcup C'))$. This means that we have an effective way to combine common attributes of the arguments of the operator $\uplus$. If c_1 is a concept name in the terminology k_1 whose definition imposes a constraint on a role R and c_2 is a concept name in the terminology k_2 whose definition constrains the role R, then $c_1 \uplus c_2$ is well-defined.

Our language has a declarative model-theoretic and a fixpoint semantics.

6 Related Work

We shortly discuss the relationship of this work to approaches regarding modeling, querying and reasoning on ontologies. The proposed frameworks and results are all different from ours.

[20] proposed an approach based on the use of rules that cross the semantic gap by creating an articulation between distributed systems. The rules are generated using a semi-automatic articulation tool with the help of a domain expert. The ontologies are represented using a graph-oriented model extended with a small set algebraic operator set. The defined algebra enables interoperation between ontologies using the articulation ontology. The input to the operators in the algebra are the ontology graphs. Unary operators like filter and extract work on a single ontology. They are analogous to the select and project in relational algebra. They help to define the interesting areas of the ontology that we want to further explore. Given an ontology and a graph pattern, an unary operation matches the pattern and returns selected portions of the ontology graph. Binary operators include union, intersection, and difference. Each

operation is defined on two ontologies and results in an ontology that can be further composed with other ontologies. Resolution of semantic heterogeneity is addressed using expert rules. Semantic relationships are represented using first-order logic based articulation rules. These rules are then modeled using the same graphical representation. In our framework, we allow richer reasoning services especially by allowing the automatic comparison of semantic carried by ontologies.

[19] proposed a merging and diagnosis tool for ontologies. The idea is to have a tool (1) able to coalesce two semantically identical terms when merging different ontologies, and (2) to identify terms that should be related by subsumption, disjointness, or instance relationships.
In [18], McGuinness presented some guidelines for conceptual modeling of ontologies in distributed environments. The guidelines are abstracted from a rich experience of the author on ontologies including, e.g., description logic applications and commercial consulting in ontologies. For example, the author notes that explicit representation of the semantics of the terms used in an ontology would be useful. In fact, this is what is considered in our framework, where terms are approximated by constraints providing a partial semantics.
In [11] a set of tools making use of the World Wide Web to enable wide access and provide users with the ability to browse, create, publish and edit ontologies stored on an ontology server are designed. For example, the browsing consists in jumping from one term to another using hyperlinks. The main goal here is to create a general environment to facilitate the development and sharing of ontologies. The tasks of exploring the relationships between ontologies and reasoning about formal statements of terms are not considered.

Finally, the combination of rule languages and description logics has been a research theme for the last decade (see, e.g., [15,9,17]). The interest from a rule language perspective came when the need was felt for manipulating hierarchically structured objects. Our approach is different in the sense that we base our language on a constraint system where individuals have a secondary role (if at all) when compared to concepts (seen as constraints).

7 Conclusion

We have overviewed a formal design for interfacing a logical query language with terminological components, having in mind to provide a declarative framework for representing, reasoning and interacting with ontologies.

We have presented a logic-based constraint language for querying ontologies and given its formal semantics. Our approach is purely declarative and formulated in terms of constraints. The implementation of query evaluation techniques requires efficient algorithms for solving the constraints. A formal account of constraint languages for ontologies is an essential step in demonstrating the correctness of such algorithms, and may yield more efficient processing strategies.

There are two interesting directions to pursue: (1) In many situations one would like to represent classes that are disjoint from each other and reason with. This means that we should be able to deal with negation in the specification language. (2) Another interesting direction of research is to significantly extend this framework to support part-whole relations [1].

References

1. Alessandro Artale, Enrico Franconi, Nicola Guarino, and Luca Pazzi. Part-Whole Relations in Object-Centered Systems: An Overview. *Data & Knowledge Engineering*, 20(3):347–383, December 1996.
2. Franz Baader. Using Automata Theory for Characterizing the Semantics of Terminological Cycles. *Annals of Mathematic and Artificial Intelligence*, 18:175–219, 1996.
3. Franz Baader and Ulrike Sattler. Number Restrictions on Complex Roles in Description Logics: A Preliminary Report. In *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA*, pages 328–339, November 1996.
4. Alexander Borgida and Peter Patel-Schneider. A Semantic and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
5. Martin Buchheit, Francesco M. Donini, and Andrea Schaerf. Decidable Reasoning in Terminological Knowledge Representation Systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.
6. B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What Are Ontologies and Why Do We Need Them. *IEEE Intelligent Systems*, 14(1):20–26, january/february 1999.
7. William W. Cohen and Haym Hirsh. Learning the CLASSIC Description Logic: Theoretical and Experimental Results. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94), Bonn, Germany*, pages 121–133, May 1994.
8. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The Complexity of Concept Languages. Technical Report RR-95-07, Deutsches Forschunggszentrum für Künstliche Intelligenz (DFKI), Kaiserslautern, Germany, June 1995.
9. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1988.
10. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in Description Logics. In *Foundation of Knowledge Representation*. Cambrige University Press, 1995.
11. Adam Farquhar, Richard Fikes, and James Rice. The Ontolingua Server: a Tool for Collaborative Ontology Construction. In *Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada*, nov 1996.
12. Nicola Guarino. *Formal Ontology in Information Systems*. IOS Press, 1998.
13. Michael Kifer and James Wu. A Logic for Object-Oriented Logic Programming (maier's o-logic revisited). In *Proceedings of the 1989 Symposium on Principles of Database Systems (PODS'89), Philadelphia, Pennsylvania*, pages 379–393, March 1989.

14. Ralf Küsters. Characterizing the Semantics of Terminological Cycles in ALN Using Finite Automata. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98), Trento, Italy*, pages 499–511, June 1998.
15. Alon Y. Levy and Marie-Christine Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
16. D. A. B. Lindberg, B. L. Humphreys, and A. T. McCray. The Unified Medical Language System. *Methods of Information in Medicine*, 1993.
17. Margarida Mamede and Luis Monteiro. A Contraint Logic Programming Scheme for Taxonomic Reasoning. In Krzysztof R. Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming (JICSLP'92), Washington, DC*, pages 255–269. MIT Press, ISBN 0-262-51064-2, 1992.
18. Deborah L. McGuiness. Conceptual Modeling for Distributed Ontology Environments. In *Proceedings of the 8th International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues, Darmstadt, Germany*, august 2000.
19. Deborah L. McGuiness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the 7th International Conference on Principles of Knowledge Representation and Reasoning (KR'2000), Breckenridge, Colorado, USA*, april 2000.
20. Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A Graph-Oriented Model for Articulation of Ontology Interdependencies. In *Proceedings of the International Conference on Extended Database Technology (EDBT'2000), Konstanz, Germany*, LNCS. Springer Verlag, march 2000.
21. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems.* LNCS-422. Springer-Verlag, 1990.
22. DA. L. Rector, W. A. Nowlan, S. Kay, C. A. Goble, and T. J. Howkins. A Framework for Modeling the Electronic Medical Record. *Methods of Information in Medicine*, 1993.
23. Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.
24. G. van Heijst, A. T. Schreiber, and B. J. Wielinga. Using Explicit Ontologies in KBS Development. *International Journal of Human and Computer Studies*, 1996.

Consistent Answers from Integrated Data Sources

Leopoldo Bertossi[1], Jan Chomicki[2], Alvaro Cortés[3], and Claudio Gutiérrez[4]

[1] Carleton University, School of Computer Science, Ottawa, Canada. bertossi@scs.carleton.ca
[2] University at Buffalo, Dept. of Computer Science and Engineering. chomicki@cse.buffalo.edu
[3] Pontificia Universidad Catolica de Chile, Departamento de Ciencia de Computación, Santiago, Chile. acortes@ing.puc.cl
[4] Universidad de Chile, Center for Web Research, Departamento de Ciencias de la Computación, Santiago, Chile. cgutierr@dcc.uchile.cl

Abstract. When data sources are integrated into a single global system, inconsistencies wrt global integrity constraints are likely to occur. In this paper, the notion of consistent answer to a global query in the context of the local-as-view approach to data integration is characterized. Furthermore, a methodology for generating query plans for retrieving consistent answers to global queries is introduced. For this purpose, an extension of the inverse-rules algorithm for deriving query plans for answering first-order queries is presented.

1 Introduction

In the last few years, due the increasing number of information sources that are available and may interact, the subject of data integration has been widely studied from different points of view. Topics like mediated schemas, query containment, answering queries using views, etc., have been considered in this context. However, the important issue of consistency of data derived from the integration process and query answering has attracted less attention.

A data integration system provides a uniform interface to several information sources. This interface, referred as *global schema* or *mediated schema*, is a context-dependent set of virtual relations used to formulate queries to the integrated system. When the user queries the system in terms of the global schema, a query processor or a mediator is in charge of rewriting the global query into a *query plan* that will eventually access the underlying information sources.

In order to perform this query rewriting, the processor needs a mapping between the mediated schema and the information sources. Two paradigms have been proposed to provide this mapping. One of them, called the *Local-as-View* (LAV) approach [15], considers each information source as a view defined in terms of relations in the global schema. The *Global-as-View* (GAV) approach considers each global predicate as a view defined in terms of the source relations [20,21].

In this paper we concentrate on the LAV approach. This scenario is more flexible than GAV for adding new data sources into a global system. Actually, preexisting data sources in the system do not need to be considered when a new source is introduced. In consequence, inconsistencies are more likely to occur. Furthermore, from the point of view of studying the logical issues around consistency of data, the LAV paradigm seems to be more challenging than the GAV. The latter is closer to the classical problem of consistency of views defined over relational databases.

In the context of the LAV approach, several algorithms have been proposed to rewrite a global query into a query plan that accesses the data source relations to answer the original query [13]. Some of them assume that certain integrity constraints (ICs) hold at the global level, and they use the ICs in the query plan generation. In [12], a rewriting algorithm that uses functional and inclusion dependencies in the mediated schema is proposed. In [8], another algorithm for query plan generation that uses functional and full dependencies is introduced. This algorithm may take a global query written in Datalog as an input. In [10], a deductive, resolution based approach to data integration is presented. It may also use global integrity constraints in the deductive derivation of the query plan. Actually, there are situations where, without assuming that certain global ICs hold, no query plan can be generated [13].

However, it is not obvious that certain desirable, global ICs will hold at the global level. After all, the data is in the sources, the global relations are virtual and there may be no consistency checking mechanism at the global level. Furthermore, and particularly in the LAV approach, it is not clear what it means for the global system to be consistent, because we do not have a global *instance*. Actually, given a set of data sources, there may be several potential global instances that (re)produce the data sources as views according to the view definitions.

A *global system* consists of a global schema and a collection of materialized data sources that are described as views over the global schema. In this context, it is quite natural to pose queries at the global level, expecting to retrieve those answers that are consistent wrt a given set of global ICs, because, as we mentioned before, global ICs may be easily violated due to the lack of a global maintenance mechanism. As the following example shows, each data source, with its own, independent maintenance mechanism, can be consistent, but inconsistencies may arise when the sources are integrated.

Example 1. Consider the global relation $R(X,Y)$ and two source relations $\{V_1(a, b),\ V_1(c,d)\}$ and $\{V_2(a,c), V_2(d,e)\}$ described by the view definitions:

$$V_1(X,Y) \leftarrow R(X,Y) \qquad V_2(X,Y) \leftarrow R(X,Y).$$

Then, the global functional dependency (FD) $R: X \rightarrow Y$ is violated, but not the local FDs $V_1:\ X \rightarrow Y$, $V_2:\ X \rightarrow Y$. □

In [3], the problem of characterizing and computing consistent query answers from a single, inconsistent relational database instance was addressed. In this case, the database instance r is considered inconsistent when it does not satisfy a given set of ICs. Intuitively speaking, an answer to a query is considered to be consistent in r if it is an answer to the query in every possible *repair* of the original instance, where a repair is a new instance that satisfies the ICs and differs from r by a minimal set of tuples under set inclusion.

However, in a global integration system we do not have an explicit database instance, but we still have an intuition of what is a consistent answer to a query posed to that system.

Example 2. (example 1 continued) If we pose to the global system the query $Q: \ Ans(X,Y) \leftarrow R(X,Y)$, we obtain the answers $\{Ans(a,b), Ans(c,d), Ans(a, c), Ans(d,e)\}$. However, only the tuples $Ans(c,d), Ans(d,e)$ should be returned as consistent answers. □

In this paper we will address the problem of posing queries to a possibly inconsistent global system, but retrieving as answers only those tuples that are consistent wrt the global ICs. To achieve this goal, we first formalize the notion of a consistent answer to a query posed to a global system. This notion is an adaptation of the notion proposed in [3] for a stand alone database. Next, we consider the problem of constructing algorithms for computing consistent answers from such a global system.

The computational mechanism proposed in [3] for a single relational database consists of rewriting the given query into a new query that, posed to the original, inconsistent database, gets as (normal) answers the consistent answers to the original query.

If we want to derive query plans for retrieving, hopefully all and only, those answers to a query posed, now, to a global integrated system, that are consistent wrt the desired, global ICs, we may try to use the approach in [3], rewriting the global query into a new query, and then pose the new query to the global system. The problem is that the rewritten query may not be handled by any of the existing query plan generation algorithms, e.g. [10,15,12], due to the presence of negation. In consequence, we need mechanisms for generating a query plan that can be applied to rewritten queries. For this purpose, we extend the "inverse-rules algorithm" from [8] to recursion-free Datalog$^\neg$ queries with built-in predicates, because this is the kind of rewritten queries we obtain. In this paper we restrict ourselves to the case of "open" sources [9], the most common scenario.

2 Preliminaries

2.1 Global Schemas and View Definitions

A *global schema* $\mathcal{R}$ is modeled by a finite set of relations $\{R_1, R_2, ..., R_n\}$ and a possibly infinite domain $\mathcal{D}$. With these relation symbols and the elements of $\mathcal{D}$ treated as constants, a first-order language $\mathcal{L}(\mathcal{R})$ can be defined. This language can be extended with new defined predicates and built-ins.

A *view*, denoted by a new predicate V, is defined by means of an $\mathcal{L}(\mathcal{R})$-formula of the form φ_V: $V(\bar{t}) \leftarrow body(\varphi_V)$, where $\bar{t}$ is a tuple containing variables and/or constants, and $body(\varphi_V)$ is a conjunction of $\mathcal{R}$-atoms. In consequence, we are considering views defined by conjunctive queries [1].

A *database instance* D over schema $\mathcal{R}$ can be considered as a first-order structure with domain $\mathcal{D}$, where the extensions (sets of tuples) of the relations R_i are finite (The extensions of built-in predicates may be infinite but fixed.). An *integrity constraint* is a first-order sentence ψ written in the language $\mathcal{L}(\mathcal{R})$. The instance D satisfies ψ, denoted $D \models \psi$, if ψ is true in D.

Given a database instance D over schema $\mathcal{R}$, and a view definition φ_V, $\varphi_V(D)$ denotes the extension of V obtained by applying the definition φ_V to D. If the view already has an extension v (given by the contents of a data source), it is possible that v is incomplete and stores only some of the tuples that satisfy the definition of V applied to D. In this case, we say the view extension v is *open* wrt D [2,9]. Most mechanisms for deriving query plans are based on open sources [15,8,18].

Following [9], we say that a *source* S is a pair $<\varphi, v>$, where φ is the view definition, and v is an extension for φ. An *open global system* $\mathfrak{G}$ (called a *source collection* in [9]), is a finite set of open sources. The schema $\mathcal{R}$ of the global system can be read from the bodies of the view definitions. It consists of the relation names that do not have a definition in the global system. The underlying domain $\mathcal{D}$ for $\mathcal{R}$ contains all the constants appearing in view extension v_i's in the sources.

When we talk about consistency in databases wrt a set of ICs we think of instances satisfying ICs. However, in a global system for data integration there is not such a global instance, at least not explicitly. Instead, a global system $\mathfrak{G}$ defines a set of legal global instances.

Definition 1. [9] *Given an open global system* $\mathfrak{G} = \{<\varphi_1, v_1>, \ldots, <\varphi_n, v_n>\}$, *the set of legal global instances is* $Linst(\mathfrak{G}) = \{D$ *instance over* $\mathcal{R} \mid v_i \subseteq \varphi_i(D),\ i = 1, \ldots, n\}$, *where* v_i *is the extension in the source* S_i *of the view defined by* φ_i, *and* $\varphi_i(D)$ *is the set of tuples obtained by applying the view definition* φ_i *to instance* D. □

Example 3. (example 2 continued) $D = \{R_1(a,b), R_1(c,d), R_1(a,c), R_1(d,e)\}$ is a legal global instance, because $v_1 = \{(a,b),(c,d)\} \subseteq \varphi_1(D) = \{(a,b),(c,d), (a,c), (d,e)\}$ and $v_2 = \{(a,c),(d,e)\} \subseteq \varphi_2(D) = \{(a,b),(c,d), (a,c),(d,e)\}$. All supersets of D are also legal global instances. □

The inverse-rules algorithm [8] for generating query plans under the LAV approach assumes that sources are open and each source relation V has a source description that defines it as a view of the global schema: $V(\bar{X}) \leftarrow P_1(\bar{X}_1), \ldots, P_n(\bar{X}_n)$. Then, for $j = 1, \ldots n$, $P_j(\bar{X}'_j) \leftarrow V(\bar{X})$ is an inverse rule for P_j. The tuple $\bar{X}_j$ is transformed to obtain the tuple $\bar{X}'_j$ as follows: if X is a constant or is a variable in $\bar{X}$, then X is unchanged in $\bar{X}'_j$. Otherwise, X is one of the variables X_i appearing in the body of the definition of V, but not in $\bar{X}$. In this case, X is replaced by a Skolem function term $f_{S,i}(\bar{X})$ in $\bar{X}'_j$. We denote the set of inverse rules of the collection $\mathcal{V}$ of source descriptions in $\mathfrak{G}$ by $\mathcal{V}^{-1}$.

Given a Datalog query Q and a set of conjunctive source descriptions in $\mathfrak{G}$, the construction of the query plan is as follows. All the rules from Q that contain global relations that cannot be defined (directly or indirectly) in terms of the global relations appearing in the source descriptions are deleted. To the resulting query, denoted as Q^-, the rules in $\mathcal{V}^{-1}$ are added; and the query so obtained is denoted by $(Q^-, \mathcal{V}^{-1})$. Notice that the global predicates can be seen as EDB predicates in the rules for Q. However, they become IDB predicates in $(Q^-, \mathcal{V}^{-1})$, because they appear in the heads of the rules in $\mathcal{V}^{-1}$. In consequence, the query plan is given essentially in terms of the source predicates.

3 Global Systems and Consistency

Intuitively speaking, *certain answers* to a query posed to a global system are those that can be obtained from every legal instance of the system. We expect *consistent answers* to the query to be a certain answers, but not necessarily the opposite will hold.

Definition 2. [2] *Given an open global system $\mathfrak{G}$ and a query $Q(\bar{X})$ to the system, a tuple $\bar{t}$ is a* certain answer *to Q in $\mathfrak{G}$ if for every global instance $D \in$ Linst$(\mathfrak{G})$, it holds $D \models Q[\bar{t}]$.* [1] □

We assume from here on that we have a fixed set of global first-order integrity constraints IC on a given global schema. We also assume that the set of ICs is consistent as a set of logical sentences. Furthermore, we will also assume that the set IC is *generic*, in the sense that IC does not determine, independently from the contents of the database, the presence/absence in/from the database of any piece of ground atomic knowledge. The ICs used in database praxis are always generic.

Definition 3. *A set IC of ICs is* generic *if there is no ground literal L in the language of IC such that $IC \models L$.* □

We need a precise definition of consistency of a global system that captures the intuitive notion of consistency related to the definitions of the sources and the data stored in the sources.

Definition 4. *Given a global system, $\mathfrak{G}$, a* minimal global instance *of $\mathfrak{G}$ is an instance $D \in$ Linst$(\mathfrak{G})$ that is minimal wrt set inclusion, i.e. there is no other instance in Linst$(\mathfrak{G})$ that is a proper subset of D (as a set of atoms). We denote by mininst$(\mathfrak{G})$ the set of minimal legal global instances of $\mathfrak{G}$ wrt set inclusion.* □

Definition 5. *A global system $\mathfrak{G}$ is* consistent *wrt to a set of global integrity constraints, IC, if every minimal legal global instance of $\mathfrak{G}$ satisfies IC: for all $D \in$ mininst$(\mathfrak{G})$, $D \models IC$.* □

[1] $D \models Q[\bar{t}]$ means that query $Q(\bar{X})$ becomes true in instance D, when tuple of variables $\bar{X}$ is assigned the values in the tuple $\bar{t}$ of database elements.

Example 4. Consider $\mathfrak{G} = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow P(X,Z) \wedge Q(Z,Y), \{V_1(a,b)\}\rangle$$

$$S_2 = \langle V_2(X,Y) \leftarrow P(X,Y), \{V_2(a,c)\}\rangle.$$

In this case, the elements of $\mathit{mininst}(\mathfrak{G})$ are of the form $D_z = \{P(a,z), Q(z,b), P(a,c)\}$ for some $z \in \mathcal{D}$. The global FD $P(X,Y)$: $X \rightarrow Y$ is violated exactly in those minimal legal instances D_z for which $z \neq c$. Thus, the global system is inconsistent. □

A global system $\mathfrak{G}$ could be inconsistent in the sense of not satisfying the given set of ICs, but still be *possible* or *realizable* in the sense that $\mathit{Linst}(\mathfrak{G}) \neq \emptyset$.

Definition 6. *(a) The ground tuple $\bar{a}$ is a minimal answer to a query Q posed to a global system $\mathfrak{G}$ if for every minimal instance $D \in \mathit{mininst}(\mathfrak{G})$, $\bar{a} \in Q(D)$, where $Q(D)$ is the answer set for Q in D.*
(b) We denote by $\mathit{Minimal}_{\mathfrak{G}}(Q)$ the set of minimal answers to query Q in $\mathfrak{G}$. □

It is clearly the case that the minimal answers contain the certain answers. For monotone queries [1], the notions of minimal and certain answers coincide. Nevertheless, in example 4 the query $\mathit{Ans}(X,Y) \leftarrow \neg P(X,Y)$ has (b,a) as a minimal answer, but (b,a) is not a certain answer, because there are legal instances that contain $P(b,a)$. Later on our queries will be allowed to contain negation. Since consistent answers have been defined relative to minimal global instances, for us the relevant notion of answer is that of minimal answer. Notice that this assumption is like imposing a form of closed-world assumption on the global instances associated with the local sources.

Given a database instance D, we denote by $\Sigma(D)$ the set of ground formulas $\{P(\bar{a}) \mid P \in \mathcal{R} \text{ and } D \models P(\bar{a})\}$.

Definition 7. [3] *(a) Let D, D' be database instances over the same schema and domain. The distance, $\Delta(D,D')$, between D and D' is the symmetric difference:*

$$\Delta(D,D') = (\Sigma(D) \setminus \Sigma(D')) \cup (\Sigma(D') \setminus \Sigma(D)).$$

(b) For database instances D, D', D'', we define $D' \leq_D D''$ if $\Delta(D,D') \subseteq \Delta(D,D'')$, i.e., if the distance between D and D' is less than or equal to the distance between D and D''. □

Definition 8. *(based on* [3]*) Let $\mathfrak{G}$ be a global system and IC a set of global ICs. A* repair *of $\mathfrak{G}$ wrt IC is a global database instance D', i.e. an instance over the global schema $\mathcal{R}$, such that:*
(a) $D' \models IC$ and
(b) D' is $\leq_D$-minimal for some $D \in \mathit{mininst}(\mathfrak{G})$. □

We can see that a repair of a global system is a global database instance that minimally differs from a minimal legal global database instance. If $\mathfrak{G}$ is consistent (definition 5), then the repairs are exactly the elements in $\mathit{mininst}(\mathfrak{G})$.

Example 5. Consider the global system $\mathfrak{G}_4 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X) \leftarrow R(X,Y), \{V_1(a)\}\rangle,$$

$$S_2 = \langle V_2(X) \leftarrow R(X,Y), \{V_2(a)\}\rangle,$$

and the global FD $R(X,Y)$: $X \rightarrow Y$. In this case, $D = \{R(a,b_1), R(a,b_2)\} \in$ *Linst*$(\mathfrak{G}_4)$, but $D \not\models IC$. However, $D \notin$ *mininst*$(\mathfrak{G}_4)$. Actually, the elements in *mininst*$(\mathfrak{G}_4)$ are of the form $\{R(a,b)\}$, for some b in the global database domain. The elements in *mininst*$(\mathfrak{G}_4)$ coincide with the repairs. Notice that $\mathfrak{G}_4$ is a consistent global system. □

Notice that in this definition of repair we are not requiring that a repair respects the property of the sources of being open, i.e. that the extension of each view in the repair contains the corresponding view extension in the source. Thus, it may be the case that a repair – still a global instance – does not belong to *Linst*$(\mathfrak{G})$. If we do not allow this, a global system might not be repairable. The notion of repair will be used as an auxiliary concept to define the notion of consistent answer.

Example 6. Consider the global system $\mathfrak{G}_5 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{V_1(a,b_1)\}\rangle,$$

$$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{V_2(a,b_2)\}\rangle,$$

and the FD $R(X,Y)$: $X \rightarrow Y$. The only element in *mininst*$(\mathfrak{G}_5)$ is $D_0 = \{R(a,b_1), R(a,b_2)\}$, that does not satisfy IC. The global system is inconsistent. The only repairs are the global instances that minimally differ from D_0 and satisfy the FD, namely $D_0^1 = \{R(a,b_1)\}$ and $D_0^2 = \{R(a,b_2)\}$. Notice that they do not belong to *Linst*$(\mathfrak{G}_5)$. □

Definition 9. *(a) Given a global system $\mathfrak{G}$, a set of global integrity constraints IC, and a global first-order query $Q(\bar{X})$, we say that a (ground) tuple $\bar{t}$ is a consistent answer to Q wrt IC, denoted by $\mathfrak{G} \models_c Q[\bar{t}]$, iff for every repair D of $\mathfrak{G}$, $D \models Q[\bar{t}]$.*
(b) We denote by $Consis_{\mathfrak{G}}(Q)$ the set of consistent answers to query Q in $\mathfrak{G}$. □

Example 7. (example 6 continued) For the query $Q_1(X) : \exists Y\ R(X,Y)$, a is a consistent answer, i.e. $\mathfrak{G}_5 \models_c \exists Y\ R(X,Y)[a]$. On the other hand, the query $Q_2(X,Y) : R(X,Y)$, does not have any consistent answer. □

Proposition 1. *Given an open global system $\mathfrak{G}$, a set of generic global integrity constraints IC, and a global query Q:*

(a) Every consistent answer is a minimal answer.
(b) If $\mathfrak{G}$ is consistent wrt IC, then every minimal answer is consistent.[2] □

[2] The proof of this proposition and of all the other explicitly stated results can be found in [5].

4 Computing Consistent Query Answers

After having given a semantic definition of consistent answer to a global query from a global system, we concentrate on the computational problem of consistent query answering (CQA).

In [3], in the context of a single, possibly inconsistent relational database, a consistent answer to a query was defined as an answer to that query that is obtained from every (minimal) repair of the database. There, for the purpose of computing consistent answers, an operator T^ω was introduced. It does the following: Given a first-order query $Q(\bar{X})$, a modified query $T^\omega(Q(\bar{X}))$ is computed. The new query $T^\omega(Q(\bar{X}))$ is posed to the original database, and the returned answers are consistent in the semantic sense.

We consider *universal* first-order integrity constraints expressed in the so-called "standard format" [3]: $\forall(\bigvee_{i=1}^{m} l_i(\bar{X}_i) \vee \varphi)$, where l_i is a database literal and φ is a formula containing built-in predicates only.

The new query $T^\omega(\varphi(\bar{x}))$ is computed by iterating an operator T which transforms an arbitrary query by appending the corresponding *residue* to each database literal appearing in the query, until a fixpoint is reached. The residue of a database literal forces the local satisfaction of the ICs for the tuples satisfying the literal. Residues can be obtained by resolution between the table and the ICs. The methodology based on the T operator is applicable to queries that are conjunctions of literals. For the usual universal ICs found in database praxis, e.g. functional dependencies, a fixpoint can be reached in a finite number of steps (see [3] for details). In the rest of this paper we will concentrate on this case.

Example 8. Consider $IC = \{R(X) \vee \neg P(X) \vee \neg Q(X),\ P(X) \vee \neg Q(X)\}$ and the query $\varphi(X) = Q(X)$. The residue of $Q(X)$ wrt the first IC is $(R(X) \vee \neg P(X))$, because if $Q(X)$ is to be satisfied, then that residue has to be true if the IC is to be satisfied too. Similarly, the residue of $Q(X)$ wrt the second IC is $P(X)$. The operator T iteratively appends the residues to the tables in the queries.

$$\begin{aligned}
T^1(\varphi(X)) &= Q(X) \wedge (R(X) \vee \neg P(X)) \wedge P(X).\\
T^2(\varphi(X)) &= T(T^1(\varphi(X))) = Q(X) \wedge (T(R(X)) \vee T(\neg P(X))) \wedge T(P(X)).\\
&= Q(X) \wedge (R(X) \vee (\neg P(X) \wedge \neg Q(X))) \wedge P(X) \wedge (R(X) \vee \neg Q(X)).\\
T^3(\varphi(X)) &= Q(X) \wedge (R(X) \vee (\neg P(X) \wedge T(\neg Q(X)))) \wedge P(X) \wedge (T(R(X)) \vee\\
&\qquad T(\neg Q(X))).
\end{aligned}$$

Since $T(\neg Q(X)) = \neg Q(X)$ and $T(R(X)) = R(X)$, we obtain $T^2(\varphi(X)) = T^3(\varphi(X))$, and a fixpoint has been reached. Since $T^\omega(\varphi(X)) := \bigwedge_{n<w} \{T^n(\varphi(X))\}$, here we obtain $T^\omega(\varphi(X)) = T^1(\varphi(X)) \wedge T^2(\varphi(X))$. □

In the context of integration of open data sources under the LAV approach, we now present an algorithm to compute consistent answers to queries that are conjunctions of literals wrt global universal integrity constraints, IC.

Given a global query $Q(\bar{X})$, the algorithm iteratively applies the T operator, and next, the rewritten query is answered by producing an appropriate query plan. A high level description of the algorithm is as follows:

Algorithm for CQA
Input: a global query Q that is a conjunction of literals.
Output: a query plan to obtain consistent answers to Q.

1. Rewrite $Q(\bar{X})$ into the first-order query $T^\omega(Q(\bar{X}))$ using IC.
2. Using a standard methodology [1,16], transform $T^\omega(Q(\bar{X}))$ into a recursion-free Datalog$^\neg$ program $\Pi(T^\omega(Q))$, possibly containing built-ins.
3. Find a query plan, $Plan(\Pi(T^\omega(Q)))$ to answer $\Pi(T^\omega(Q))$ seen as a query to the global system.
4. Evaluate the query plan on the view extensions of $\mathfrak{G}$ to compute a set of answers.

Program $\Pi(T^\omega(Q))$ may contain negation. There are currently no mechanisms to obtain query plans for global queries containing negation (although some heuristics are sketched in [10]). On the positive side, the generated Datalog$^\neg$ program does not contain recursion. $Plan(\Pi(T^\omega(Q)))$ is a new Datalog$^\neg$ query program that is run on top of the data sources that act as the extensional database.

Example 9. Consider the global system $\mathfrak{G}_8 = \{S_1, S_2\}$, with

$$S_1 = \langle V_1(X,Y) \leftarrow R(X,Y), \{V_1(a,b)\}\rangle$$

$$S_2 = \langle V_2(X,Y) \leftarrow R(X,Y), \{V_2(a,c), V_2(c,d)\}\rangle,$$

the FD $R(X,Y)$: $X \rightarrow Y$, and the global query $Q(X,Y)$: $R(X,Y)$.

Here, the only element in $mininst(\mathfrak{G}_8)$ is $D_0 = \{R(a,b), R(a,c), R(c,d)\}$. The only minimal instance violates FD through the tuples $[a,b]$, $[a,c]$. In consequence, the global system $\mathfrak{G}_8$ is inconsistent. It has two repairs, $D_0^1 = \{R(a,b), R(c,d)\}$ and $D_0^2 = \{R(a,c), R(c,d)\}$. The only consistent answer to the query is the tuple $[c,d]$.

We now apply the Algorithm for CQA. First of all, we need the FD expressed in a first-order language, it becomes $\forall XYZ(R_1(X,Y) \wedge R_1(X,Z) \rightarrow Y = Z)$. At step 1. the query has to be rewritten. For functional dependencies, the operator T has a finite fixpoint and we obtain

$$T^\omega(Q(X,Y))\colon\ R(X,Y) \wedge \neg\exists Z(R(X,Z) \wedge Z \neq Y). \tag{1}$$

This query can also be translated into the following Datalog$^\neg$ program $\Pi(T^\omega(Q))$:

$$Ans(X,Y) \leftarrow R(X,Y),\ not\ S(X,Y)$$

$$S(X,Y) \leftarrow R(X,Z), Y \neq Z.$$

We need a query plan to answer this query program containing negation. □

5 Plans for Queries with Negation

The inverse-rules algorithm in [8] for generating a query plans for a global Datalog query Q produces an answer set that is *maximally-contained* in the answer set for Q. In [17], it is shown that such a maximally-contained query plan retrieves all certain answers to the query Q.

A limitation of the inverse-rules algorithm for query plans is that it allows Datalog queries only. In our case, even if we start with a conjunctive global query, the query program obtained after transforming the query rewritten using the T operator may contain negation. In consequence, we need to find plans for recursion-free Datalog$^\neg$ query programs with built-in predicates, like the $\Pi(T_w(Q))$'s. Notice that the absence of recursion immediately makes the resulting query programs stratified [1].

Now, we present an extension of the algorithm in [8] to the case where the query program contains negation, but no recursion.

Given a recursion-free Datalog$^\neg$ query Q in terms of global and defined relations, the extended inverse-rules algorithm works analogously to the one presented in [8], except that in the case a rule in Q contains a negated literal in the body, say $S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), \neg G(\bar{X}), L_n(\bar{X}), \ldots$., it is checked if G can be eventually evaluated in terms of the global relations appearing in the source descriptions (because those are the global relations that can be eventually evaluated using the inverse rules). If this is not the case, then that goal is eliminated, obtaining the modified rule $S(\bar{X}) \leftarrow \ldots, L_1(\bar{X}), L_n(\bar{X}), \ldots$. We show the methodology by means of an example[3].

Example 10. Consider the global system $\mathfrak{G}_9 = \{S_1, S_2\}$ with

$$S_1 = \langle V_1(X,Z) \leftarrow R_1(X,Y), R_2(Y,Z), \{V_1(a,b)\}\rangle$$
$$S_2 = \langle V_2(X,Y) \leftarrow R_3(X,Y), \{V_2(c,d)\}\rangle.$$

and the global query Q:[4]

$$Ans(X,Z) \leftarrow R_1(X,Y), R_2(Y,Z),\ not\ R_4(X,Y)$$
$$R_4(X,Y) \leftarrow R_3(X,Y),\ not\ R_5(X,Y) \qquad (2)$$
$$R_7(X) \leftarrow R_1(X,Y), R_6(X,Y). \qquad (3)$$

The inverses rules $\mathcal{V}^{-1}$ are obtained from the source descriptions:

$$R_1(X, f_1(X,Z)) \leftarrow V_1(X,Z)$$
$$R_2(f_1(X,Z), Z) \leftarrow V_1(X,Z)$$
$$R_3(X,Y) \leftarrow V_2(X,Y).$$

[3] It should be easy to extend this methodology to stratified Datalog$^\neg$ queries, but we do not need this extension here.

[4] This example, containing implicit existential quantifiers, is used to illustrate the extended inverse-rules algorithm only. In this paper we are not dealing with the problem of computing consistent answers to existentially quantified conjunctive global queries.

To compute a query plan for Q, we first need Q^-:

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z),\ \mathit{not}\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y). \end{aligned}$$

The literal *not* $R_5(X,Y)$ was eliminated from rule (2) because it does not appear in any source description. For the same reason (the literal $R_6(X,Y)$ does not appear in any source description), rule (3) was eliminated. Then, the query plan $(Q^-, \mathcal{V}^{-1})$ is:

$$\begin{aligned} Ans(X,Z) &\leftarrow R_1(X,Y), R_2(Y,Z),\ \mathit{not}\ R_4(X,Y) \\ R_4(X,Y) &\leftarrow R_3(X,Y) \\ R_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) \\ R_2(f_1(X,Z),Z) &\leftarrow V_1(X,Z) \\ R_3(X,Y) &\leftarrow V_2(X,Y). \end{aligned}$$

Finally, the query plan can be evaluated in a bottom-up manner starting from the sources V_1, V_2 to retrieve $Ans(a,b)$ as final answer for the global query Q. Tuples at the answer level that contain Skolem function symbol f_1 can be pruned from the answer set. □

Now we will show that the generated plan can be finitely evaluated in a bottom-up manner, and that the query plan is maximally contained in the query.

5.1 Containment

In this section we will establish that the resulting query plan is maximally contained in the original query. We will concentrate on recursion free Datalog$^\neg$ programs (including built-ins).

Given a query plan P for Q using $\mathcal{V}$, the *expansion* P^{exp} of P is obtained from P by replacing all source relations in P with the bodies of the corresponding views $\mathcal{V}$, and using fresh variables for existential variables in the views. That is, the source relations are eliminated by reinverting the inverse rules, in order to be in position to compare the original query, expressed in terms of global predicates, and the query plan.

Example 11. (example 10 continued) To compute P^{exp}, we first rename the IDB predicates in P.

$$\begin{aligned} Ans(X,Z) &\leftarrow P_1(X,Y), P_2(Y,Z),\ \mathit{not}\ P_4(X,Y) \\ P_4(X,Y) &\leftarrow P_3(X,Y) \\ P_1(X, f_1(X,Z)) &\leftarrow V_1(X,Z) && (4) \\ P_2(f_1(X,Z),Z) &\leftarrow V_1(X,Z) && (5) \\ P_3(X,Y) &\leftarrow V_2(X,Y). && (6) \end{aligned}$$

The expansion is the following Datalog$^\neg$ query P^{exp}:

$$\begin{aligned}
Ans(X,Z) &\leftarrow P_1(X,Y), P_2(Y,Z),\ \mathit{not}\ P_4(X,Y)\\
P_4(X,Y) &\leftarrow P_3(X,Y)\\
P_1(X, f_1(X,Z)) &\leftarrow R_1(X,U), R_2(U,Z)\\
P_2(f_1(X,Z),Z) &\leftarrow R_1(X,U), R_2(U,Z)\\
P_3(X,Y) &\leftarrow R_3(X,Y).
\end{aligned}$$

□

Remark 1. Notice that in the extended plan obtained in the previous example, we could keep the rules (4), (5), (6), plus the source definitions in S_1, S_2 of example 10. Sometimes we will do this in the rest of this section.

Theorem 1. *For every recursion free Datalog$^\neg$ program Q, every set of conjunctive source descriptions $\mathcal{V}$, and all finite instances of the source relations, the query plan $(Q^-, \mathcal{V}^{-1})$ has a unique finite minimal fixpoint. Furthermore, bottom-up evaluation is guaranteed to terminate, and produces this unique fixpoint.* □

Definition 10. *(maximally-contained plan* [17]*) A query plan P is maximally contained in a query Q using views $\mathcal{V}$ if $P^{exp} \subseteq Q$ and for every query plan P' such that $(P')^{exp} \subseteq Q$, it holds $P' \subseteq P$.*[5] □

If $S = \{v_1, \ldots, v_m\}$ is a set of extensions for the sources, and P is a query plan, then $P(S)$ denotes the extension of the query predicate Ans, obtained by evaluating P on S. Notice that $P(S)$ may contain tuples with Skolem function symbols. We denote by $P(S)\!\downarrow$ the set of tuples in $P(S)$ that do not contain function symbols, and by $P\!\downarrow$ the plan that computes this pruned extension.

In order to prove maximal containment, the notion of proof tree presented in [19] for Datalog queries can be extended to recursion free Datalog$^\neg$ queries (see [5]).

Theorem 2. *(compare* [8]*, thm. 8) For every recursion free Datalog$^\neg$ program Q and every set of conjunctive source descriptions $\mathcal{V}$, the query plan $(Q^-, \mathcal{V}^{-1})\downarrow$ is maximally contained in Q.* □

The previous theorem can be restricted to the case of minimal global instances (see [5] for details). That is, the query plan $(Q^-, \mathcal{V}^{-1})\downarrow$ is contained in Q, i.e. $(Q^-, \mathcal{V}^{-1})^{exp}(D) \subseteq Q(D)$ for every minimal global instance D (this is an immediate consequence of the theorem, that holds for arbitrary global instances); and every other plan, whose expansion is contained in Q for minimal global instances, is also contained in $(Q^-, \mathcal{V}^{-1})\downarrow$ (this follows from a particular minimal global instance D constructed in the proof). As a consequence, we obtain the following theorem.

Theorem 3. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, and a recursion free Datalog$^\neg$ query Q with built-ins, the plan $Plan(Q)$ obtained with the extended inverse-rules algorithm retrieves exactly $Minimal_{\mathfrak{G}}(Q)$.* □

[5] Here, $P^{exp} \subseteq Q$ means that the extension of the query predicate in P^{exp} in included in the extension of Q for every database instance over the global schema.

6 The CQA Algorithm Revisited

We have presented an algorithm for generation of query plans for global queries that are Datalog$^\neg$ queries without recursion.

Example 12. (example 9 continued) Applying the extended inverse-rules methodology we obtain the following query plan $Plan(\Pi(T^\omega(Q)))$:

$$\begin{aligned} Ans'(X,Y) &\leftarrow R(X,Y),\ \textit{not}\ S(X,Y) && (7)\\ S(X,Y) &\leftarrow R(X,Z), Y \neq Z && (8)\\ R(X,Y) &\leftarrow V_1(X,Y)\\ R(X,Y) &\leftarrow V_2(X,Y). \end{aligned}$$

This query plan can be evaluated on the view extensions in $\mathfrak{G}_8$ either top-down or bottom-up. In the latter case, a domain predicate for variable Y in rule (8) can be used, that due to the first goal in (7) can be defined as containing the values of the second attribute of R.[6] In either case, the only answer obtained is $Ans'(c,d)$. This answer is consistent: $\mathfrak{G}_8 \models_c R(X,Y)[c,d]$. Notice that, instead of this query program, the original query (1) could be evaluated using SQL2, defining first R as a view that is the union of V_1 and V_2. □

Proposition 2. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, the consistent answers to a conjunctive global query Q correspond to $Minimal(T^\omega(Q))$. Furthermore, if $T^\omega(Q)$ is monotone, then its certain answers are the consistent answers to Q.* □

Theorem 4. *Given an open global system $\mathfrak{G}$, IC a set of generic ICs, and a conjunctive global query Q, $Plan(\Pi(T^\omega(Q))$ retrieves exactly $Consis_{\mathfrak{G}}(Q)$.* □

7 Conclusions

We have concentrated on open sources. In the future, it would interesting to extend the semantics of consistent query answers to the case where the global system contains also *closed* and *clopen* sources. The extension of the algorithm presented in this paper to those cases is left for future work. The work in [9] introduces a useful framework to deal with those more general global systems.

The methodology we have presented here is based on the methodology presented in [3]. In consequence, it applies to a limited class of queries and constraints. Other approaches to consistent query answering from a single database are based on logic programs with stable model semantics [4,6,11]. They can handle general first-order queries. It would be interesting to explore how the methodology presented there can be integrated with the methodology presented in this paper in order to consistently answer queries posed to global integrated systems under the LAV approach. Consistency issues under the GAV approach are treated in [14].

[6] A general solution consists in defining such a domain as containing the (appropriate) data values stored in the sources.

Acknowledgments. Work supported by FONDECYT Grant 1000593, NSF Grant INT-9901877/CONICYT Grant 1998-02-083, NSF Grant IIS-0119186, Carleton University Start-Up Grant 9364-01, NSERC Grant 250279-02, Nucleus Millennium for Web Research (Mideplan, Grant P01-029-F). We are grateful to Alberto Mendelzon for stimulating and useful conversations.

References

1. Abiteboul, S.; Hull, R. and Vianu, V. *Foundations of Databases.* Addison-Wesley, 1995.
2. Abiteboul, A. and Duschka, O. Complexity of Answering Queries Using Materialized Views. In *Proc. 9th Annual ACM Symp. on the Theory of Computing*, ACM Press, 1998, pp. 254-263.
3. Arenas, M.; Bertossi, L. and Chomicki, J. Consistent Query Answers in Inconsistent Databases. In *Proc. ACM Symposium on Principles of Database Systems (ACM PODS'99)*, ACM Press, 1999, pp. 68–79.
4. Arenas, M.; Bertossi, L. and Chomicki, J. Specifying and Querying Database Repairs Using Logic Programs with Exceptions. In *Flexible Query Answering Systems. Recent Developments.* H. Larsen, J. Kacprzyk, S. Zadrozny, H. Christiansen (eds.), Springer, 2000, pp. 27–41.
5. Bertossi, L.; Chomicki, J.; Cortes, A. and Gutierrez, C. Consistent Answers from Integrated Data Sources. Extended version. May 2002. http://www.scs.carleton.ca/~bertossi/papers/paperleo14.ps
6. Barcelo, P. and Bertossi, L. Repairing Databases with Annotated Predicate Logic. In *Proc. Ninth International Workshop on Non-Monotonic Reasoning (NMR'2002). Special session on Changing and Integrating Information: From Theory to Practice.* S. Benferhat and E. Giunchiglia (eds.), Morgan Kaufmann Publishers, 2002, pp. 160 – 170.
7. Cali, A.; Calvanese, D.; De Giacomo, G. and Lenzerini, M. Data integration Under Integrity Constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, Springer LNCS 2348, 2002, pp. 262–279.
8. Duschka, O.; Genesereth, M. and Levy, A. Recursive Query Plans for Data Integration. *Journal of Logic Programming*, 2000, 43(1):49-73.
9. Grahne, G. and Mendelzon, A. Tableau Techniques for Querying Information Sources through Global Schemas. In *Proc. of the Int. Conf. on Database Theory (ICDT'99)*, Springer LNCS 1540, 1999, pp. 332–347.
10. Grant, J. and Minker, M. A Logic-based Approach to Data Integration. *Theory and Practice of Logic Programming*, 2002, 2(3):323-368.
11. Greco, G.; Greco, S. and Zumpano, E. A Logic Programming Approach to the Integration, Repairing and Querying of Inconsistent Databases. In *Proc. 17th International Conference on Logic Programming (ICLP'01)*, Ph. Codognet (ed.), Springer LNCS 2237, 2001, pp. 348–364.
12. Gryz, J. Query Rewriting Using Views in the Presence of Functional and Inclusion Dependencies. *Information Systems*, 1999, 24(7):597–612.
13. Halevy, A. Answering Queries using Views: A Survey. *VLDB Journal*, 2001, 10(4):270–294.
14. Lembo, D.; Lenzerini, M. and Rosati, R. Source Inconsistency and Incompleteness in Data Integration. In *Proc. of the 9th International Workshop Knowledge Representation meets Databases (KRDB'02)*, 2002. http://CEUR-WS.org/Vol-54/

15. Levy, A.; Rajaraman, A. and Ordille, J. Querying Heterogeneous Information Sources using Source Descriptions. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB'96)*, Morgan Kaufmann Publishing Co., 1996, pp. 251–262.
16. Lloyd, J. *Foundations of Logic Programming.* Springer, 1987.
17. Millstein, T.; Levy, A. and Friedman, M. Query Containment for Data Integration Systems. In *Proc. of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'00)*, ACM Press, 2000, pp. 67–75.
18. Pottinger, R. and Levy, A. A Scalable Algorithm for Answering Queries Using Views. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, Morgan Kaufmann Publishing Co., 2000, pp. 484–495.
19. Ramakrishnan, R.; Sagiv, Y.; Ullman, J.D. and Vardi, M.Y. Proof-tree Transformation Theorems and their Applications. In *Proceedings ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, (PODS'89)*, ACM Press, 1989, pp. 172–181.
20. Ullman, J. Information Integration using Logical Views. In *Proc. of the Int. Conf. on Database Theory (ICDT'97)*, Springer LNCS 1186, 1997, pp. 19–40.
21. Ullman, J.D. Information Integration Using Logical Views. *Theoretical Computer Science*, 2000, 239(2):189-210.

A Knowledge-Based Query System for Biological Databases

Paolo Bresciani[1] and Paolo Fontana[2]

[1] ITC-irst, via Sommarive, 18, I-38050 Trento-Povo, Italy
bresciani@itc.it
[2] Istituto Agrario di S.Michele all'Adige, via Mach 1
S. Michele a/Adige, TN, Italy
pfontana@itc.it

Abstract. In more than one decade, genomic research produced a huge amount of experimental data. Although these data are usually freely available on the World Wide Web, accessing them in a consistent and fruitful way is not always an easy task. One of the main causes of this problem can be recognized in the lack of user interfaces that are sufficiently flexible and, at the same time, highly interactive and cooperative and also semantically precise.
This paper tackles this issues by presenting a semantics drive paradigm for formulating queries to genomic and protein databases. The paradigm is founded on knowledge-based reasoning capabilities, in order to provide the user with a semantics driven guide in the difficult task of building the intended query.

1 Introduction

Bioinformatics is the science of storing, extracting, organizing, analyzing, interpreting, and utilizing biological information. Genomic research —in particular, advances in DNA sequencing and genome mapping techniques— has provided huge amounts of data for populating several single-organism databases (e.g., [5,26,17]), as well as databases on many species [24,4,25]. Thanks to the decreasing cost of gene sequencing techniques, data on genetic sequences and the related databases are exponentially increasing.

If on the one hand, and for several years, the primary goal of genomic research and bioinformatics has been to make experimental data and resources merely available in order to allow for subsequent possible data interpretation and scientific models and theories construction, on the other hand the capability of managing such kind of data at the best —for storing, extracting, organizing, analyzing, and make them easily usable and accessible— is becoming one of the crucial factors for the next developments in bioinformatics.

Nevertheless, up to now, only few efforts have been made aimed at proposing appropriate tools for accessing distributed and syntactically heterogenous genomic and protein databases. Of course, the efficient and effective access to this kind of databases involves different challenges when various aspects of information technologies are dealt with. Although some of these topics have been tackled, such as the design of appropriate query languages [22] that allow for a better functional interoperability and integration [21] of distributed and syntactically heterogeneous genomic databases, still only few efforts have been made in order to provide flexible, clever, and user-friendly interfaces, capable of drastically easing the task of query formulation.

Form based query interfaces like that provided with the Sequence Retrieval System (SRS) [16] are currently the most commonly used by the biologists community, although they allow only a very little flexibility, and in most of the cases are inadequate and make query formulation difficult, when not an impossible, as evidenced in [21]. On the other hand, the use of generic query languages like *SQL*, even if extended to better fit the need of accessing multiple genomic databases, like in the case of *GQL* [22], is viable only for those biologists that acquired some expertise with them.

This paper proposes a novel paradigm for the realization of easy, intuitive, and flexible user-interfaces for accessing complex and structured biological databases.

This is a crucial aspect, because the realization of intuitive interfaces that really help the user represents a necessary condition to gain user's acceptance and confidence in the provided data and knowledge resources.

The paradigm is based on the use of knowledge representation tools and ontologies for flexible and intuitive formulation of queries against relational and object-oriented biological databases.

The present paper is organized as follows. Section 2 introduces users needs in accessing databases on complex domains. Section 3 present our knowledge based paradigm for driving intelligent query interfaces. Finally, Section 4 present some technical details on the implementation of the reasoning services. Conclusions are given in Section 5.

2 User Needs and Query Systems

The task of building a query for accessing data stored in a database can be tackled in different ways by different kinds of user. In any case, the query must be expressed in a form suitable to be processed by the system or the specific application, and, at the same time, it must have a clear meaning for the user.

Users can be classified on the basis of their familiarity with databases, their knowledge of the application domain, the frequency of use of the application, and the heterogeneity and complexity of the queries they formulate [13,14]. They may range from *expert* users, with a good understanding of the notions underling the data models for databases design and a good knowledge of the specific application domain, down to *naïve* users, with no background in data modeling techniques and tools, no knowledge about database design and implementation, and only a minimal knowledge of the application domain. They are likely to use the application only for simple and predefined tasks.

In the case of genomic and protein databases, users can generally be considered as domain expert with little or no expertise neither in database technologies, nor in formalisms and methods for conceptual modeling. Thus, they are placed in a particularly unfavorable position: although they need to formulate sophisticated and extemporary queries, they are likely to have no familiarity with structured query languages, and possibly they are just occasional users, unwilling to spend many efforts in trying to understand the way the information is logically organized in the different and many databases.

The so called “Visual Query Systems” (VQS) [14] try to overcome these difficulties by providing graphical paradigms for visualizing the conceptual model of the available data, and by allowing to build queries by means of visual interactions. The VQS are, typically, addressed to all those situations in which: (i) the application domain is complex, and, therefore, a simple form-based interface, or even a QBE-like interface [27] —as, e.g., that of SRS— is insufficient; (ii) the user is eager to discover interesting information and to derive knowledge from the highly structured set of available data.

Nevertheless, most VQS still fail to adequately support the users in the task of understanding the *semantics* of the data at an intensional level. In fact, although they typically provide graphical representations of the data models, they scarcely support the users in strictly linking models with their intuitive understanding of the domain scenario. In order to better understand this connection, the users must perform a set of trials, during which they can *discover* the real meaning of the stored data by analyzing and comparing different answers to different tentative queries.

Moreover, the VQS typically require a good understanding of the diagrammatic notations used by database design methodologies, like, e.g., Entity Relationship.

In the presence of a really complex domains, like the bio-molecular ontologies, the situation is even harder. In fact, in these cases, even a user with a good understating of the data models and with a good knowledge of the scenario may find difficulties in combining a necessarily imperfect knowledge of the domain and an almost missing vision of the connection between it and the logical structure of the database, on the one hand, with the necessity of building sophisticated queries, on the other hand. For example, it can be difficult to avoid inconsistent requests, or noticing similarities between apparently different queries. This can lead to two kinds of problems:

- in her attempts of building the intended query, the user can fail several times, because reiterating slightly different, but all inconsistent, queries;
- it is possible that, after having submitted a query to a database, the answer is unexpectedly too large or too narrow: in this case, trying to modify it in a structural and relevant way could be an hard task.

For example, consider a simple case where the user initially submits a query specifying that she wants to retrieve the records on all the proteins that are expressed in the cell nucleus and have *KDEL receptor* function. Of course no record will be returned, because these proteins are only present in the endoplasmic reticulum (see, e.g., [23]). To keep our example very simple, we can imagine that, once the user somehow realizes the non-sense of her query, she tries to formulate a new query, slightly generalizing the first one by replacing the function *KDEL receptor* with a more generic function, for example *endoplasmic reticulum receptor*. A standard interface cannot suggest to the user that nothing better has been done (the query still is inconsistent), and database access time (and possibly money) is wasted again. Moreover, the only information obtained by the user, and only after having actually submitted the query, is that no record of the required type is present in the database; but nothing prevent the user from thinking that —maybe in the future or in other databases— such a record could be available. Thus, further vain searches may be done elsewhere or in other occasions.

Of course, the example sketched above is trivial for any biologist, but with slightly more specialized terms it may be easily transformed into a real problem even for expert biologists. In these cases, a tool that could exhibit an intuitive semantic description of the database content would be of great value.

For these reasons, we propose here a semi-visual paradigm, where more emphasis is put on knowledge based services, rather than on diagrammatic visualizations. In our paradigm, query validation and comparison can be carried out at the intensional level.

The idea is to consider the task of formulating the intended query as an iterative process, possibly requiring several steps of refinement and modification depending on the responses of the system. During this process, the user is assisted in *understanding* the semantics of the queries progressively built. Also, a classification of the queries with respect to the set of classes defined in the conceptual model of the database is constantly provided, giving a way for progressively understanding and learning the semantics of the model itself. The classification capability allows us to immediately verify if a query has inconsistent semantics with respect to the database conceptual model.

The resulting process of query formulation is a highly interactive one, and uses an *intensional* representation of the query. That is, the user works on the query itself, and not on the data (the *extension*) that may be in the answer to the query; only when the query is considered to be well formulated it is submitted to the database. Thus, there is a much higher chance to get the desired results. A consequence is that the process results in a faster and cheaper (in the case of pay-per-record) access, especially to remote or multiple databases.

The proposed paradigm uses, in order to convey the conceptual model and query meaning to the user, a uniform graphical presentation. But its most important characteristic is that, being based on a semantic representation of the database conceptual model, it allows us to perform some relevant inferential tasks on the semantics of the queries. In particular, it allows the user to:

1. interactively and iteratively build queries;
2. be prevented from building inconsistent queries;
3. interactively explore the semantics of the queries and of the classes involved;
4. be gradually introduced only to those parts of the conceptual model that are relevant for the query formulation;
5. be provided with simple, but effective, features for query refinement and query generalization.

3 The Knowledge Based Interaction

To exhibit the behavior and the features mentioned in the previous section, a system implementing our paradigm must provide a classification of the query with respect to the set of classes that form the database conceptual model. In particular, classifying the query at each step of the interaction allows us to immediately detect semantic inconsistencies.

Thus, it is then necessary that the query corresponds to an object description, possibly with nested relationships with other queries. Therefore, such kind of descriptions must be dynamically generated and managed. From the graphical point of view, a query can be represented as in Figure 1 (in the left part). The example shown in Figure 1 corresponds to the following query:

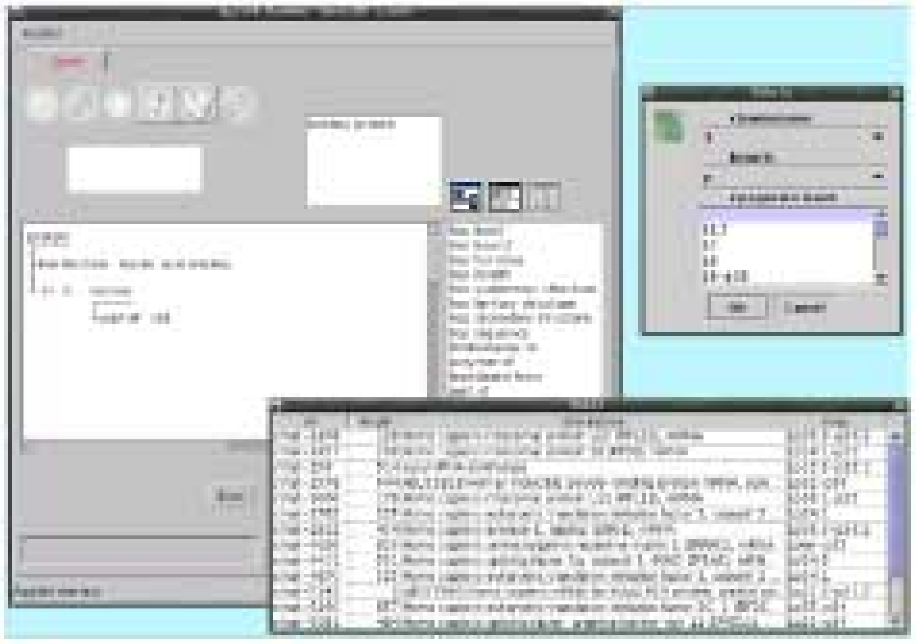


Fig. 1. The interface proposed by our paradigm, as appears in our implemented prototype. The picture include also the interface for answer browsing.

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{has-function}(x,w) \wedge \texttt{Nucleic-Acid-Binding}(w) \wedge \\ \texttt{is-in}(x,y) \wedge \texttt{Nucleus}(y) \wedge \texttt{part-of}(y,z) \wedge \texttt{Cell}(z).$$

We believe that the graphical notation in Figure 1 is quite easily grasped by the unskilled user, because it is quite close to a natural language *Noun Phrase*, but, at the same time, avoids referential ambiguities. Moreover, the representation shows only the necessary terms of the class hierarchy, and avoids to disorientate the user with the too many details usually provided by most of the VQS interfaces.

At the same time, the user is allowed to interactively build sophisticated queries simply by choosing among the classes defined in the conceptual model, and iteratively forming, with them, complex boolean expressions of literals connected by links representing relationships. In Subsection 3.1 the main functionalities for building queries will be presented. An important aspect is that the query that is going to be built is constantly guaranteed to be consistent with respect to the conceptual model of the database. Moreover, the query is classified with respect to the conceptual model after each modification, so that the user can be informed about the position of the query in the conceptual taxonomy.

Requiring that only consistent queries are built prevents the user from uselessly querying the database. This is the minimal service that a semantics driven interface should provide. Moreover, our paradigm assumes also other constraints on the interactions allowed for building the query. Altogether, it is required that:

- Only *consistent* actions (i.e., actions that lead to a new query that still is consistent) are allowed.
- Only *relevant* modifications (i.e., query transformations that lead to new queries that are semantically not equivalent to the original one) are proposed.
- Only *close* modifications (i.e., not all the consistent and relevant modifications, but only those that lead to a new query with semantics close to the original query semantics) are proposed. This constraint is required in order to avoid to overload the user with too many options.[1]

In the following we shortly call these constraints "CRCP", that stands for "Consistent, Relevant, and Close Property".

In the next subsection the principal kinds of interactions are listed.

3.1 Query Building

We implemented the features illustrated above in a prototypical concept demonstrator [8], the interface of which is shown in Figure 1. The window on the left includes all the element for building the query:

1. an area where to represent the query;
2. a list of names of classes more generic than the query (let's call it a *generalization list*);
3. a list of names of classes more specific than the query (*specialization list*, hidden in the picture);
4. a list of names of classes equivalent to the query (*equivalence list*);
5. the list of relationships that can be used to build or modify the query;
6. a set of buttons that allow us to perform various actions for building the query.

The picture also shows how attributes can be restricted (window at the upper right of the screen snapshot) and a possible way to present the results (window at the bottom).

The prototype has been configured in order to access the "Muscle-TRAIT" biological DB. The Muscle-TRAIT DB [1] is part of a project carried on at CRIBI Biotech Center of the University of Padua, aimed at identifying and characterizing genes expressed in human skeletal muscle. The initial project was based on systematic sequencing of ESTs. The most relevant data are stored in the TRAIT DB (TRAnscript Integrated Table). The DB is periodically updated with cross-references to LocusLink [24], a database maintained at NCBI (National Center for Biotechnology Information) where, when possible, entries are tagged with references to the biological ontology *GeneOntology* [2]. Thus, every Muscle-TRAIT entry that has an homology with a human sequence in LocusLink inherits the association with one or more terms of GeneOntology.

The interaction with the prototype starts with the selection of a first term among those proposed in an initial list, that, in the specific configuration, are protein, biological function, and cell part. After that, the user can iteratively modify the query by the following functionalities:

[1] This point is one of the most delicate. In fact, it is difficult, in general, to give a clear notion of *close* query and *close* modification. Moreover, in some cases, some overload could be acceptable, or even desirable, if it provides for some shortcut in the interaction.

- *Query replacement:* that allows for the selection of an initial term in the taxonomy, and for the substitution of the whole query with an atomic term semantically close to it (more general, specific, or equivalent).
- *Relation selection:* that allows to restrict the query by imposing the existence of relationships with other atomic terms or complex expressions.
- *Propositional combination with other terms:* that allows for the boolean combination of selected parts of the query with other terms.
- *Negation:* that allows for the negation of literals.
- *Refinement:* that allows for the replacement of selected subexpressions, instead of the boolean combination.

The buttons in the interface give access to these function.

An example of use of the concept demonstrator and the above functionalities is described in [8]. In Subsection 4.3 a different example is introduced in abstract terms.

4 Reasoning Services

As mentioned, the main requirement of our paradigm is that each of the functionality listed above must comply the CRCP. To obtain this behavior some reasoning capabilities are needed. We obtain these capabilities by representing the conceptual model of the database and the query itself in terms of a Description Logic Knowledge Base.

In fact, it has been show that Description Logics (DL) offer powerful formalisms for solving several problems concerning data modeling and access [6,12], like, for example, schemata integration and, in particular, intelligent query management [11,7].

Below, we briefly recall some basic notions on DL, and exemplify how conceptual models and queries can be represented. Then, it will be possible to introduce the reasoning services that allow to guarantee the CRCP for the functionalities listed in section 3.1.

4.1 Description Logics

Description Logics are, essentially, variable-free concise reformulations of decidable restricted fragments of First Order Logic (FOL). The syntax of DL allows to express *concepts* (unary predicate symbols), *roles* (binary predicate symbols), and *individuals*[2] (constants). In different DL slightly different languages are used, with different levels of expressiveness. In all of them [15] concept expressions and role expressions can be built starting from atomic concepts and roles, using different sets of operators, like the concept-forming operators: $\sqcap, \sqcup, \neg, \forall, \exists$, and the role-forming operators: $\cdot^{-1}, \circ$.

Description logics semantics can be given, for example, by mapping DL expressions into FOL formulæ [11]: an atomic concept `A` and an atomic role `P` are mapped —or *interpreted*— into the FOL open atomic formulæ $A(\gamma)$ and $P(\alpha, \beta)$. In table 1, for some DL concept expressions C and D the corresponding FOL open formulæ $F_C(\gamma)$ and $F_D(\gamma)$ are recursively given; similarly, for some DL role expressions R and Q the corresponding FOL open formulæ $F_R(\alpha, \beta)$ and $F_Q(\alpha, \beta)$ are given.

[2] Reasoning about individuals (the *assertional reasoning*) is not relevant for our purposes; therefore, only the *terminological* aspects of DL will be considered.

Table 1. FOL transformational semantics for some DL operators.

Construct	FOL Semantics
A	$A(\gamma)$
$(\neg C)$	$\neg F_C(\gamma)$
$(C \sqcap D)$	$F_C(\gamma) \wedge F_D(\gamma)$
$(C \sqcup D)$	$F_C(\gamma) \vee F_D(\gamma)$
$(\forall R.C)$	$\forall x.F_R(\gamma, x) \Rightarrow F_C(x)$
$(\exists R.C)$	$\exists x.F_R(\gamma, x) \wedge F_C(x)$
$\vdots$	$\vdots$
P	$P(\beta, \alpha)$
(R^{-1})	$F_R(\beta, \alpha)$
$(R \circ Q)$	$\exists x.F_R(\alpha, x) \wedge F_Q(x, \beta)$
$\vdots$	$\vdots$

An important feature of DL is that they are equipped with a formal calculus that allows to perform some inferential tasks [15]. The most relevant, here, are (i) *consistency checking*: a concept description C is said to be *consistent* iff the corresponding FOL formula, $F_C(\gamma)$, is satisfiable; (ii) *subsumption*: a concept description C is said to *subsume* a concept description D (written $D \sqsubset C$) iff $\forall x.F_D(x) \Rightarrow F_C(x)$ is a FOL tautology.

In DL it is possible to define *knowledge-bases* (KB) as sets of subsumption assertions. It is said that a subsumption is entailed by a knowledge base ($KB \models D \sqsubset E$) iff $F_{KB} \models \forall x.F_C(x) \Rightarrow F_D(x)$ in terms of FOL.[3] The consistency of a concept description C with respect to a (consistent) KB can be expressed as: $KB \models C \not\equiv \bot$.[4]

4.2 Conceptual Modeling in DL

To see how a database conceptual model can be described in terms of Description Logics, let's consider the fragment of knowledge base (from now on, the "KB") shown in Figure 2, encoding the TRAIT DB conceptual model and domain.[5]

Using KB allows the system to infer, for example, that whenever a *Protein* is located (*is-in*) in the *Nucleus*, it *has-funct* only of type *Nucleic-Acid-Binding* or *Transcription-Factor-Binding*, and that in this case *Binding* is a good summarization of both, and that saying $\neg Cytosol$ as range of *is-in* is equivalent to saying *Nucleus*, and so on.

[3] Where F_{KB} is the set of FOL axioms containing $\forall x.F_{C_1}(x) \Rightarrow F_{C_2}(x)$ for each $C_1 \sqsubset C_2$ in KB.

[4] $C \equiv D$ stands for $C \sqsubset D \wedge D \sqsubset C$, and the special concept $\bot$ correspond to the symbol denoting the *false* in FOL. Similarly, the special concept $\top$ corresponds to the symbol denoting the *truth* in FOL.

[5] Of course there are many different possible encodings; the one in Figure 2 tries to be the most readable, although not the shortest one.

$$\begin{array}{l}
\mathit{Protein} \sqcap \mathit{Cell\text{-}Part} \equiv \bot \\
\mathit{Protein} \sqcap \mathit{Function} \equiv \bot \\
\mathit{Function} \sqcap \mathit{Cell\text{-}Part} \equiv \bot \\[1ex]
\exists \mathit{has\text{-}funct}.\top \sqsubset \mathit{Protein} \\
\exists \mathit{has\text{-}funct}^{-1}.\top \sqsubset \mathit{Function} \\[1ex]
\exists \mathit{is\text{-}in}.\top \sqsubset \mathit{Protein} \\
\exists \mathit{is\text{-}in}^{-1}.\top \sqsubset \mathit{Cell\text{-}Part} \\[1ex]
\mathit{Protein} \sqsubset \forall \mathit{has\text{-}funct}.\mathit{Function} \sqcap \exists^{\geq 1} \mathit{has\text{-}funct} \sqcap \forall \mathit{is\text{-}in}.\mathit{Cell\text{-}Part} \sqcap \exists^{\geq 1} \mathit{is\text{-}in} \\
\mathit{Function} \sqsubset \forall \mathit{has\text{-}funct}^{-1}.\mathit{Protein} \sqcap \exists^{\geq 1} \mathit{has\text{-}funct}^{-1} \\[1ex]
\mathit{Nucleus} \sqsubset \mathit{Cell\text{-}Part} \\
\mathit{Cytosol} \sqsubset \mathit{Cell\text{-}Part} \\
\mathit{Nucleus} \sqcap \mathit{Cytosol} \equiv \bot \\
\mathit{Cell\text{-}Part} \sqsubset \mathit{Nucleus} \sqcup \mathit{Cytosol} \\[1ex]
\mathit{Binding} \sqsubset \mathit{Function} \\
\mathit{Nucleic\text{-}Acid\text{-}Binding} \sqsubset \mathit{Binding} \\
\mathit{Transcription\text{-}Factor\text{-}Binding} \sqsubset \mathit{Binding} \\[1ex]
\mathit{Binding\text{-}Protein} \equiv \mathit{Protein} \sqcap \exists \mathit{has\text{-}funct}.\mathit{Binding} \\[1ex]
\mathit{Protein} \sqcap \exists \mathit{has\text{-}funct}.\mathit{Nucleic\text{-}Acid\text{-}Binding} \sqsubset \forall \mathit{is\text{-}in}.\mathit{Nucleus} \\
\mathit{Protein} \sqcap \exists \mathit{has\text{-}funct}.\mathit{Transcription\text{-}Factor\text{-}Binding} \sqsubset \forall \mathit{is\text{-}in}.\mathit{Nucleus} \\
\mathit{Protein} \sqcap \exists \mathit{is\text{-}in}.\mathit{Nucleus} \sqsubset \\
\qquad \forall \mathit{has\text{-}funct}.(\mathit{Nucleic\text{-}Acid\text{-}Binding} \sqcup \mathit{Transcription\text{-}Factor\text{-}Binding}) \\
\quad \vdots
\end{array}$$

Fig. 2. Translation of the conceptual model into a KB.

4.3 DL Reasoning and the CRCP

In other papers [10,8] we motivated our approach by presenting further details and examples on the interaction. In this last section, instead, we shortly introduce the reasoning services that are implemented in our concept demonstrator.

Thus, consider that the intended query to be built is, for example:[6]

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg\texttt{Cytosol}(z) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Function}(y) \tag{1}$$

One possible path followed for building it could first go through the formulation of the following query:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Function}(y)$$

and then add to it the further constraint:

$$\texttt{is-in}(x,z) \wedge \neg\texttt{Cytosol}(z)$$

[6] From now on, for editing reasons, we write the queries in they linear form including variables. Of course the system present them in the graphical/textual notation shown in the Figure 1.

Once obtained (1), the user may want to refine the term `Function`. The system will then evidence that `Function` may be replaced by `Binding` without changing the semantics. The capability of showing such a kind of *contextual* equivalence is important, because this conveys to the user more knowledge about the conceptual model.

If instead the user tried to build query (1) by starting from the query:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg\texttt{Cytosol}(z)$$

and then adding the constraint on `has-funct`, the system would immediately propose the following completion:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg\texttt{Cytosol}(z) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Binding}(y) \tag{2}$$

In fact, $\texttt{Binding}(y)$ is the most specific constraint applicable to y without introducing loss of generality. Thus, form (2) of the query is also the most informative.

The behavior exhibited by the system is a consequence of the CRCP listed in section 3, considering the fact that the two queries (1) and (2) are equivalent in the context of the conceptual schema of the database.

The features illustrated above, and others, are obtained by means of some reasoning services that can be implemented in terms of the basic inferential services of *consistency checking* and *subsumption* provided by a DL reasoner. In order to do this, it is necessary to represent the query in DL. For example, query (1) is represented by:

$$Protein \sqcap \exists is\text{-}in.\neg Cytosol \sqcap \exists has\text{-}funct.Function \tag{3}$$

And query (2) by:

$$Protein \sqcap \exists is\text{-}in.\neg Cytosol \sqcap \exists has\text{-}funct.Binding \tag{4}$$

It is quite easy to verify that the equivalence among term (3) and term (4) is entailed by the knowledge base ($KB \models (1) \equiv (2)$).

Of course, this is just one example of reasoning: in particular it is relevant for the functionality of *relation selection* seen in subsection 3.1. Another example, also visualized in Figure 1, is:

$$Protein \sqcap \exists is\text{-}in.(Nucleus \sqcap \exists part\text{-}of.Cell) \sqcap \exists has\text{-}funct.Nucleic\text{-}Acid\text{-}Binding$$

$$\sqsubset$$

$$Binding\text{-}Protein$$

More systematically, the kinds of checks corresponding to each of the functionalities introduced in subsection 3.1 are sketched below.

Query replacement. It is enough to maintain a representation of the query in term of DL, like in expression (3), and to *classify* it in the *taxonomy* implied by the knowledge base. That is, the most specific subsumers of Q —$MSS(Q)$— and the most generic subsumees of Q —$MGS(Q)$— must be found in the KB, in order to update the generalization and the specialization lists, respectively.

Relation selection. Here the task is more complex, and require two steps:

1. list all the *compatible relationships* by testing, for each relation name R in the KB, if $Q \sqcap \exists R.\top$ is consistent (similarly, for the inverse relation, the test is: $Q \sqcap \exists R^{-1}.\top \not\equiv \bot$);
2. for each compatible relationship R (or R^{-1}) determined with step 1, define the relevant range as $MSS(\exists R^{-1}.Q)$ (or $MSS(\exists R.Q)$).

Propositional combination. Let's consider the case of AND combinations. Assume that q is the sub-query, selected in Q, for which a list of possible (i.e., CRCP compliant) terms to be conjuncted is sought. Let $Q_{q/q'}$ denote the new query obtained from Q by replacing the sub-query q with q' in Q. That is, what must be checked is the CRCP of $Q_{q/(q \sqcap C_i)}$ with respect to Q, for all the $C_i \in KB$. It is worth noticing that only the specialization list is meaningful during this operation. Several strategies can be adopted in order to reduce the search of the C_i, as, for example, following a top-down search in the taxonomy, with early pruning when inconsistent cases are found, and no further search when a relevant C_i is reached (in this case $Q_{q/(q \sqcap C_i)}$ will be the closest to Q, along the considered path, due the top-down search). The top-down strategy gives good results because, in practical cases, the conceptual model contains disjoint classes at a quite high level in the taxonomy, allowing for a good amount of pruning.

For OR combinations the general approach is dual, but the dual (bottom-up) strategy is not as efficient as the top-down strategy for the conjunctions.

Negation. The case of negation is quite easy: it is enough to check if the query ($Q_{q/\neg q}$) will still be consistent.

Refinement. In this case the situation is similar to the case of propositional combination, seen above, but a bidirectional search must be performed. The CRCP must be checked on all the possible Q_{q/C_i}. A possible strategy is to start from $MSS(q)$ and $MGS(q)$, and search upward and downward respectively, in order to find the generalization and the specialization lists. Also in this case, reaching a relevant C_i or a C_i that makes Q_{q/C_i} inconsistent halts the search along the considered path.

Of course, the paradigm here proposed can be implemented only in presence of a DL automatic reasoner, capable of managing KB with circular definitions (or general axioms). It is also clear that the DL language required must allow to express conjunctions, disjunctions, negations, qualified quantifications and number restrictions, and inverse roles. $\mathcal{SHIQ}$ has all these characteristics. In our implementation we used the DL reasoner iFaCT for the $\mathcal{SHIQ}$ Description Logics [20]. The MSS and the MGS services, as well as all the checks and functionalities described above, have been implemented by us over iFaCT. The KB representing the conceptual model and domain of the Muscle TRAIT DB has been built by us starting from TAMBIS [3] and GeneOntology [2]. More details on these aspects and on the architecture of the prototype can be be found in [8].

5 Conclusion and Discussion

We have introduced a novel paradigm for providing knowledge based flexible query interfaces to complex databases. Our paradigm may be applied in several applications domains [9,10], whenever: (i) the application domain is intrinsically complex; (ii) the queries the user wants to build are likely to be structurally complex and extemporary; (iii) the user is eager to discover, for better formulating the intended query, the structure of the conceptual model, through an interactive and iterative process. In particular, in this paper, the paradigm has been illustrated in the context of biological databases.

The main advantage of our approach lies in the fact that the user can be supported, by classification and consistency checking services, in iteratively and incrementally formulating the intended query, with a minimal cognitive effort and a minimization of malformed or unwanted queries.

In the last part of this paper we focused, in particular, on the definition of the reasoning services that must be provided so that the paradigm can be fully implemented. They are soundly based on the use of the iFaCT reasoner for the highly expressive $\mathcal{SHIQ}$ Description Logics. Reasoning with expressive Description Logics is known to be an hard problem in general [15], and, in particular, for $\mathcal{SHIQ}$ it is EXPTIME-complete. Fortunately, the pathological cases are also the most artificial. Several optimization techniques are implemented in iFaCT [18] and for some practical examples there is empirical evidence that they can provide a considerable speed-up [19].

In the case of our prototypical implementation we still have to optimize it and scale it up to verify its applicability in a more real scenario. Indeed, our future research plans include the collaborations with bio-technology institutes to thoroughly experiment our approach with the collaborations of biologists who may act as test-users.

Acknowledgements. Among others, a special thank goes to Giorgio Valle and Stefano Toppo for having kindly provided the TRAIT DB. As well, we thank Ian Horrocks for having made iFaCT available.

Paolo Fontana also thanks University of Trento as well as IASMAA and Fondazione Cassa di Risparmio di Trento e Rovereto for financing his research.

References

1. http://muscle.cribi.unipd.it.
2. M. Ashburner, C. A. Ball, J. A. Blake, H. Butler, J. M. Cherry, J. Corradi, K. Dolinski, J. T. Eppig, M. Harris, D. P. Hill, S. Lewis, B. Marshall, C. Mungall, L. Reiser, S. Rhee, J. E. Richardson, J. Richter, M. Ringwald, G. M. Rubin, G. Sherlock, and J. Yoon. Creating the gene ontology resources: design and implementation. *Genome Research*, 11(8):1425–1433, Aug. 2001.

3. P. G. Baker, C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bionformatics*, 15(6):510–520, 1999.
4. D. A. Benson, D. J. L. Ilene Karsch-Mizrachi, J. Ostell, B. A. Rapp, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 30(1):17–20, 2002.
5. J. A. Blake, J. E. Richardson, C. J. Bult, J. A. Kadin, J. T. Eppig, and T. M. G. D. Group. he mouse genome database (mgd): the model organism database for the laboratory mouse. *Nucleic Acids Research*, 30(1):113–115, Jan. 2002.
6. A. Borgida. Description logics for data management. *IEEE Transactions on Knowledge and Data Engineering*, 5(7), Oct. 1995.
7. P. Bresciani. The challenge of integrating knowledge representation and databases. *Informatica*, 20(4):443–453, Dec. 1996.
8. P. Bresciani, P. Fontana, and P. Busetta. A knowledge based interface for distributed biological databases. In D. Marra, E. Merelli, P. Romano, and G. Rossi, editors, *Proceedings of the NETTAB international workshop on Agents in Bioinformatics*. NETTAB, University of Bologna, July 2002.
9. P. Bresciani, M. Nori, and N. Pedot. A knowledge based paradigm for querying databases. In *Proceedings of the 11th International Conference and on Database and Expert Systems Applications (DEXA 2000)*, volume 1873 of *Lecture Notes in Artificial Intelligence*, pages 794–804. Springer, Sept. 2000.
10. P. Bresciani, M. Nori, and N. Pedot. QueloDB: a knowledge based visual query system. In *Proceeding of the 2000 International Conference on Artificial Intelligence (IC-AI 2000), volume III*, Las Vegas, June 2000. CSREA Press.
11. M. Buchheit, M. A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.
12. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1998.
13. T. Catarci, S. Chang, M. Costabile, S. Levialdi, and G. Santucci. A graph-based framework for multiparadigmatic visual access to databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):455–475, 1996.
14. T. Catarci, M. F. Costabile, S. Levialdi, and C.Batini. Visual query systems for databases: a survey. *Journal of Visual Languages and Computing*, 8(2):215–260, Apr. 1997.
15. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the* 2^{nd} *International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162, Cambridge, MA, 1991.
16. T. Etzold and P. Argos. Srs —an indexing and retrieval tool for flat file data libraries. *Comput Appl Biosci*, 9(1):49–57, Feb. 1993.
17. FlyBase Consortium. The FlyBase database of drosophila genome project and community literature. *Nucleic Acids Research*, 27:85–88, 1999.
18. I. Horrocks. Reasoning with expressive description logics: Theory and practice. In A. Voronkov, editor, *Proc. of the 18th Int. Conf. on Automated Deduction (CADE-18)*, number 2392 in Lecture Notes in Artificial Intelligence, pages 1–15. Springer-Verlag, 2002.
19. I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th Eur. Conf. on Artificial Intelligence (ECAI 2002)*, pages 277–281, July 2002.
20. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
21. H. M. Jamil. Achieving interoperability of genome databases through intelligent web mediator. In *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000)*, Washington DC, Nov. 2000. IEEE.

22. H. M. Jamil. Gql: A reasonable complex sql for genomic databases. In *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000)*, Washington DC, Nov. 2000. IEEE.
23. B. Lewin. *GenesVII*. Oxford University Press, 2000.
24. K. D. Pruitt and D. R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Research*, 29(1):137–140, 2001.
25. G. Stoesser, W. Baker, A. van den Broek, E. Camon, M. Garcia-Pastor, C. Kanz, T. Kulikova, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, N. Redaschi, P. Stoehr, M. A. Tuli, K. Tzouvara, and R. Vaughan. The EMBL nucleotide sequence database. *Nucleic Acids Research*, 30(1):21–26, 2002.
26. M. Westerfield, E. Dorrey, A. E. Kirkpatrik, and A. Douglas. Zebrafish informatics and ZFIN the database. *Methods Cell Biol.*, 60:339–355, 1999.
27. M. M. Zloof. Query-by-example: A database language. *IBM System Journal*, 16(4):324–343, 1977.

On Measuring Similarity for Conceptual Querying

Henrik Bulskov, Rasmus Knappe, and Troels Andreasen

Department of Computer Science,
Roskilde University,
P.O. Box 260, DK-4000 Roskilde, Denmark
{bulskov,knappe,troels}@ruc.dk

Abstract. The focus of this paper is approaches to measuring similarity for application in connection with query evaluation. Rather than only comparing at the level of words the issue here is to compare concepts that appear as compound expressions derived from list of words through brief natural language analysis. Concepts refers to and are compared with respect to an ontology describing the domain of the database. We discuss three different principles for measuring similarity between concepts. One in the form of subsumption expansion of concepts and two as different measures of distance in a graphical representation of an ontology.

1 Introduction

The magnitude and complexity of available information sources constitutes a rapidly increasing conglomerate, that requires both a general overview and domain specific insight knowledge in order to query and utilize.

While classical term based text retrieval systems produce answers of a relatively high quality, considering the simplicity of the methods involved, there definitely appears to be room for improvement. Especially when dealing with the motivating problem in classical term based text retrieval systems – their obvious inability to interpret queries as anything other than a list of terms. This constitutes a problem because the semantics of the text fragment is lost along with the context and the semantic relations between concepts [7].

We are looking for improvement that can enhance the systems ability to generate ideal answers.

1.1 The OntoQuery Project

The foundation for the work in this paper is the interdisciplinary research project OntoQuery[1] (Ontology-based Querying)[3,4,10]. The primary goal for the OntoQuery project is to contribute to the development of theories and methodologies for content-based text retrieval from text databases.

[1] The project has the following participating institutions: Centre for Language Technology, The Technical University of Denmark - Informatics and Mathematical Modelling, Copenhagen Business School - Data linguistics, Roskilde University - Intelligent Systems Laboratory and the University of Southern Denmark.

This is sought done by introducing:

- A formal description language, ONTOLOG [9], whose expressions functions as descriptions for concepts in the domain texts, in the queries and in the ontology.
- A method for doing ontology-based linguistic analysis.
- Theories for ontology-based query processing that efficient compares descriptions of queries with descriptions of elements in the text domain [2,1].

Along with the work on theoretical issues a prototype system with a set of accompanying tools is developed for validation and demonstration of the theoretical results [5].

2 Object Representation in the OntoQuery Project

An important aspect in developing information retrieval systems concerns the comparing of queries with the objects held by the system.

One approach is a common representation, for both queries and objects, in which we bring the description of the query and the description of the objects on a directly comparable form.

A central aspect of this approach is that descriptions are created as intermediate representations of "*content*". A description is a set of descriptors describing a text fragment. For a text fragment, e.g. a sentence, a simple form description expresses the content by means of a nested set of words from the sentence.

Descriptions have the general form:

$$D = \{D_1, \ldots, D_n\} = \{\{D_{11}, \ldots, D_{1m}\}, \ldots, \{D_{n1}, D_{n2}, \ldots, D_{nm}\}\}$$

where each descriptor D_i is a set of concepts $D_{i1}, \ldots, D_{im}$. This representation is shown in figure 1.

$$\begin{tikzcd}
 & & D_{11} \\
 & D_1 \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & D_{12} \\
D \arrow[ur] \arrow[r] \arrow[dr] \arrow[ddr] & D_2 & \vdots \\
 & \vdots & D_{1m} \\
 & D_n &
\end{tikzcd}$$

Fig. 1. The nested set representation.

To generate descriptions, text fragments are prepared by a parser that employ the knowledge base. The parser can in principle scale from a simple word recognizer to a complex natural language parser that maps the full meaning content of the sentence into an internal representation.

Since the issue here is IR the idea is of course to grab fragments of content rather than represent full meaning, and the building stones are concepts. The structure of descriptions should be understood considering this aim.

The approach to description generation in the OntoQuery project is a subject to ongoing development. In the present state descriptions are generated as follows.

A tagger identifies heuristically categories for words. Based on tags and a simple grammar, a parser divides the sentence by framing identified noun phrases (NPs) in the sentence. For each part of the sentence, corresponding to an NP, a descriptor is produced as a set of concepts.

Take as an example the sentence:

"Physical well-being caused by a balanced diet"

A description, consisting of nouns and adjectives, in a simple form without taking into account the framing of NP's in the sentence could be:

(physical), (well-being), (balanced), (diet)

With the framing of NP's the describers can be gathered giving the description:

(physical well-being), (balanced diet)

2.1 Introducing Semantic Relations

The representation described until now, and implemented in the present prototype in the OntoQuery project, is nested sets of concepts, as shown in Fig. 1. But not all the information from the linguistic analysis is mapped into this representation. For instance when representing "*balanced*, *diet*" as the set of two words {*balanced*, *diet*} the information about the semantic relation (obvious *characterized by*) is not preserved.

The experiments with the present prototype has shown that the nested set representation is suitable for describing the complexity of sentences and is efficient as model for the query evaluations.

Changing the innermost level in the nested sets from concept to semantic expressions, will preserve the properties of the nested set representation and at the same time increase the expressive power.

In the sentence *"Physical well-being caused by a balanced diet"* the nouns (e.g. concepts) are *well-being* and *diet* and the words indicating the semantic relation between these concepts are *caused by*. Without the semantic relation this sentence describes something about *well-being* and *diet*, but not exactly how they are connected. The semantic relations combine the concepts, constituting a more complex concept. In this example the concept of physical well-being caused by a balanced diet.

The semantic relation in the above example is *caused by.* Natural language has an arbitrary set of semantic relations but in the IR context only a minor subset is relevant [8]. In examples in this paper we use the subset of semantic relations shown in Table 1

Table 1. The subset of semantic relations and their abbreviations used in this paper.

Semantic relations
caused by (CBY)
characterized by (CHR)
concept inclusion (ISA)
with respect to (WRT)

2.2 Semantic Expressions

A representation with nested sets of semantic expression needs a formal language to describe compound concepts formed by semantic relations.

ONTOLOG is a concept algebra for integration, formalization, representation and reasoning with semantics of natural language and ontologies.

Expressions in ONTOLOG are descriptions of concepts situated in an ontology formed by an algebraic lattice with concept inclusion as the ordering relation.

The basic elements in ONTOLOG are concepts and binary relations between concepts. The algebra introduces two closed operations on concept expressions φ and ψ [9]:

- conceptual *sum* $(\varphi + \psi)$, interpreted as the concept being φ or ψ
- conceptual *product* $(\varphi \times \psi)$, interpreted as the concept being φ and ψ

Relationships r are introduced algebraically by means of a binary operator (:), the Peirce product $(r : \varphi)$, which combines a relation r with an expression φ, resulting in a expression, thus relating nodes in the ontology.

The typical use of the Peirce products is as a factor in conceptual products, as in $c \times (r\colon c_1)$, which can be rewritten to form the feature structure $c[r\colon c_1]$, where $[r\colon c_1]$ is an attribution of the concept c.

As an example consider the sentence "disorders caused by hormones". One possible resulting description could be:

$$disorder[CBY\colon hormones]$$

Nesting is supported as in:

$$disorder[CBY\colon lack[WRT\colon vitaminD]]$$

Describing a "disorder caused by lack of vitamin D". A concept c with multiple attributions is denoted:

$$c\begin{bmatrix} r_1\colon \varphi_1 \\ \dots \\ r_n\colon \varphi_n \end{bmatrix}$$

Which translates into the ONTOLOG expression $c \times r_1(\varphi_1) \ldots \times r_n(\varphi_n)$.

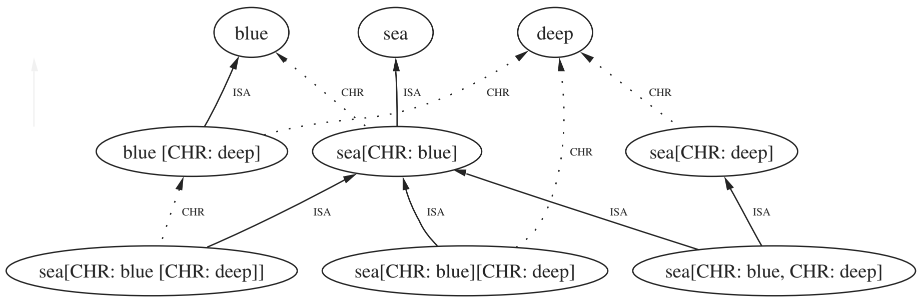


Fig. 2. A visualization of different semantic interpretations of the expression "*deep blue sea*".

2.3 Visualizing Ontolog Expressions

Ontolog expression can be translated into directed graphs, by analyzing the semantics of the expression. An Ontolog expression, O consists of a set of compound concepts C, a set of (directed) semantic relations R and a mapping $\delta\colon R \to C \times C$, relating compound concepts.

Figure 2 is the visualization of the expression "*deep blue sea*", which has three different semantic interpretations:

- *sea*[CHR: *blue*, CHR: *deep*], the sea which is blue and deep.
- *sea*[CHR: *blue*][CHR: *deep*], the blue sea which is deep.
- *sea*[CHR: *blue* [CHR: *deep*]], the sea whose (color) is deep blue.

In Ontolog we have only two different interpretations because the *sea*[CHR: *blue*, CHR: *deep*][2] and *sea* [CHR : *blue*] [CHR : *deep*] are equal due to the associative nature of Ontolog.

3 From Ontology to Similarity

In building a query evaluation principle that draws on an ontology, a key issue is of course how the ontology influence the matching of values, that is, how the different relations of the ontology may contribute to similarity. We have to decide for each relation to what extent related values are similar and we must build similarity functions, mapping values into similarities, that reflect these decisions.

We discuss firstly below how to introduce similarity based on the key ordering relation in the ontology: hyponomy relation[3] as applied on atomic concepts

[2] The expression *sea*[CHR: *blue*, CHR: *deep*] is equivalent to *sea* $\begin{bmatrix} CHR\colon blue \\ CHR\colon deep \end{bmatrix}$.

[3] Concept inclusion (ISA).

(concepts explicitly represented in the ontology). Secondly we discuss how to extend the notion of similarity to cover – not only atomic but – general compound concepts as expressions in the language ONTOLOG. This involves the semantic relations introduced above as part of the language and as complementing the primary hyponomy relation.

3.1 Similarity on Atomic Concepts

In the OntoQuery project the hyponomy or concept inclusion relation plays a central role as the ordering relation that bind the ontology in a lattice. Concept inclusion intuitively imply strong similarity in the opposite direction of the inclusion (specialization), but also the direction of the inclusion (generalization) must contribute with some degree of similarity. Take as an example the small fraction of an ontology in figure 3. With reference to this ontology the atomic concept *dog* can be directly expanded to cover also *poodle* and *alsatian*.

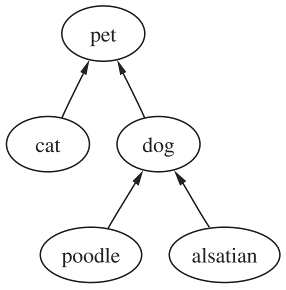


Fig. 3. Inclusion relation (ISA) with upwards reading, e.g. dog ISA animal.

This expansion respects the ontology in the sense that every instance of the extension of the expanded concept *dog* (that is, every element in the union of the extensions of *dog*, *poodle* and *alsatian*) by definition bear the relation *ISA* to *dog*. The intuition is that to a query on *dog* an answer including instances *poodle* is satisfactory (a specific answer to a general query). Since the hyponomy relation obviously is transitive we can by the same argument expand to further specializations e.g. to include *poodle* in the extension of *animal*. However similarity exploiting the lattice should also reflect 'distance' in the relation. Intuitively greater distance (longer path in the relation graph) corresponds to smaller similarity.

Further also generalization should contribute to similarity. Of course it is not strictly correct in an ontological sense to expand the extension of *dog* with instances of *animal*, but because all *dogs* are *animals*, *animals* are to some degree similar to *dogs*. This substantiates that also a property of generalization similarity should be exploited and, for similar reasons as in the case of specializations, that also transitive generalizations should contribute with decreasing degree of similarity.

A concept inclusion relation can be mapped into a similarity function in accordance with the described intuition as follows.

Assume an ontology reflecting domain knowledge and comprising a partial ordering *ISA*-relation. Figure 3 shows an example. Apart from the references shown as edges a number of implicit references, for instance *poodle ISA animal*, can be derived because of the transitivity. To make "distance" influence similarity we need either to be able to distinguish explicitly stated, original references from derived or to establish a transitive reduction of the *ISA* relation. Let $ISA^{\sim}$ be a stated or derived non-transitive relation such that *ISA* becomes the transitive closure of $ISA^{\sim}$. Similarity reflecting distance can then be measured from path-length in the $ISA^{\sim}$ lattice. A similarity function *sim* based on distance in $ISA^{\sim}$ $dist(X,Y)$ should have the properties:

1. sim: $U \times U \rightarrow [0,1]$, where U is the universe of concepts
2. $sim(X,Y) = 1$ only if $X = Y$
3. $sim(X,Y) < sim(X,Z)$ if $dist(X,Y) < dist(X,Z)$

By parameterizing with two factors δ and γ expressing similarity of immediate specialization and generalization respectively, we can define a simple similarity function: If there is a path from nodes (concepts) X and Y in the hyponomy relation then it has the form

$$P = (P_1, \cdots, P_n) \text{ where } P_i ISA^{\sim} P_{i+1} \text{ or } P_{i+1} ISA^{\sim} P_i \text{ for each } i \tag{1}$$

with $X = P_1$ and $Y = P_n$.

Given a path $P = (P_1, \cdots, P_n)$, set $s(P)$ and $g(P)$ to the numbers of specializations and generalizations respectively along the path P thus:

$$s(P) = \left|\left\{ i |\ P_i\ ISA^{\sim}\ P_{i+1} \right\}\right| \text{ and } g(P) = \left|\left\{ i |\ P_{i+1}\ ISA^{\sim}\ P_i \right\}\right| \tag{2}$$

If $P^1, \cdots, P^m$ are all paths connecting X and Y then the degree to which Y is similar to X can be defined as

$$sim(X,Y) = \max_{j=1,\ldots,m} \left\{ \sigma^{s(P^j)} \gamma^{g(P^j)} \right\} \tag{3}$$

This similarity can be considered as derived from the ontology by transforming the ontology into a directional weighted graph, with σ as downwards and γ as upwards weights and with similarity derived as the product of the weights on the paths. An atomic concept T can then be expanded to a fuzzy set, including T and similar values $T_1, T_2, \ldots, T_n$ as in:

$$T+ = 1/T + sim(T,T_1)/T_1 + sim(T,T_2)/T_2 + \cdots sim(T,T_n)/T_n \tag{4}$$

Thus for instance with $\sigma = 0.9$ and $\gamma = 0.4$ the expansion of the concepts *dog*, *animal* and *poodle* into sets of similar values would be:

$dog+ = 1/dog + 0.9/poodle + 0.9/alsatian + 0.4/animal$
$poodle+ = 1/poodle + 0.4/dog + 0.36/alsatian + 0.16/animal + 0.144/cat$
$animal+ = 1/animal + 0.9/cat + 0.9/dog + 0.81/poodle + 0.81/alsatian$

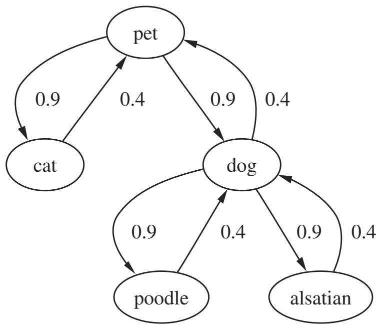


Fig. 4. The ontology transformed into a directed weighted graph, with the immediate specialization and generalization similarity being $\sigma = 0.9$ and $\gamma = 0.4$ respectively as weights. Similarity is derived as maximal (multiplicative) weighted path length, thus $sim(poodle, alsatian) = 0.4 * 0.9 = 0.36$.

3.2 General Concept-Similarity

The semantic relations, used in forming concepts in the ontology, indirectly contribute to similarity through subsumption. For instance *disorder*[CBY: *lack* [WRT: *vitaminD*]] is subsumed by - and thus extensionally included in - each of the more general concepts *disorder*[CBY: *lack*] and *disorder*. Thus with a definition of similarity covering atomic concepts, and in some sense reflecting the ordering concept inclusion relation, we can extend to similarity on compound concepts by a relaxation, which takes subsumed concepts into account when comparing descriptions.

The principle can be considered to be a matter of subsumption expansion. Any compound concept is expanded (or relaxed) into the set of subsuming concepts, thus

$$disorder[CBY: lack[WRT: vitaminD]]$$

is expanded to the set

$$\{disorder, disorder[CBY: lack], disorder[CBY: lack[WRT: vitaminD]]\}$$

One approach to query-answering in this direction is to expand the description of the query along the ontology and the potential answer objects along subsumption.

For instance a query on *disease* could be expanded to a query on similar values like:

$$disease+ = 1/disease + \ldots + 0.4/disorder + \ldots$$

and a potential answer object like $disorder[CBY: lack[WRT: vitaminD]]$ would then be expanded as exemplified above.

While not the key issue here, we should point out the importance of applying an appropriate averaging aggregation when comparing descriptions. It is essential that similarity based on subsumption expansion, exploits that for instance the degree to which $c[r_1 : c_1]$ is matching $c[r_1 : c_1[r_2 : c_2]]$ is higher than the degree for c with no attributes is matching $c[r_1 : c_1[r_2 : c_2]]$. Approaches to aggregation that can be tailored to obtain these properties, based on order weighted averaging[11] and capturing nested structuring[12], are described in [1,2].

An alternative to the above described subsumption expansion could be to generalize the principle of weighted path similarity as described in the previous subsection for the ISA-relation. While the similarity between c and $c[r_1 : c_1]$ can be claimed to be justified by the ontology formalism (subsumption) or simply by the fact that $c[r_1 : c_1]$ ISA c, it is not strictly correct in an ontological sense to claim similarity likewise between c_1 and $c[r_1 : c_1]$.

For instance $disorder[CBY : lack]$ is conceptually not some kind of a $lack$. On the other hand it would be reasonable to claim that $disorder[CBY : lack]$ in a broad sense has something to do with (and thus has similarities to) $lack$. Most examples tend to reveal the same characteristics and this phenomenon is one good explanation for the comparative success of conventional word-based querying approaches. Basically the (incorrect) assumption of no correlation between words in NL phrases, which is underlying any strictly word-based approach, does not lead to serious failure because the correlation that appears is not dominating.

This could of course be an argument for not looking at compound concepts at all, but rather these considerations points in the direction of redrawing some of the importance of correlation in NL phrases when developing similarity measures.

Consider figure 5. The solid edges are ISA references and the broken are references by other semantic relations – in this example CBY and WRT are in use. When ignoring the broken edges, the graph shown is a subgraph of the key lattice of the ontology (where ISA is the ordering relation). Each compound concept has broken edges to its attribution concept. The spelling out of the full compound concept expression as the label of a node is redundant (the concept expression can be derived from the connecting edges).

The principle of weighted path similarity can be generalized by introducing similarity factors for the semantic relations. The extensional arguments used to argue for differentiated weights depending on direction does not apply to semantic relations and seemingly there is no obvious way to differentiate based on direction at all. Thus one approach in the generalization is simply to introduce a single similarity factor and to transform to bidirectional edges.

Assume that we have k different semantic relations $R^1, \ldots, R^k$ and let $\rho_1, \cdots, \rho_k$ be the attached similarity factors. Given a path $P = (P_1, \cdots, P_n)$, set $r^j(P)$ to the number of R^j edges along the path P thus:

$$r^j(P) = \left|\left\{ i \middle| \; P_i \; R^j \; P_{i+1} \right\}\right| \tag{5}$$

If $P^1, \cdots, P^m$ are all paths connecting X and Y then the degree to which Y is similar to X can be defined as

$$sim(X,Y) = \max_{i=1,\ldots,m} \left\{ \sigma^{s(P^i)} \gamma^{g(P^i) r^1(P)} \cdots {\rho_k}^{r^k(P)} \right\} \tag{6}$$

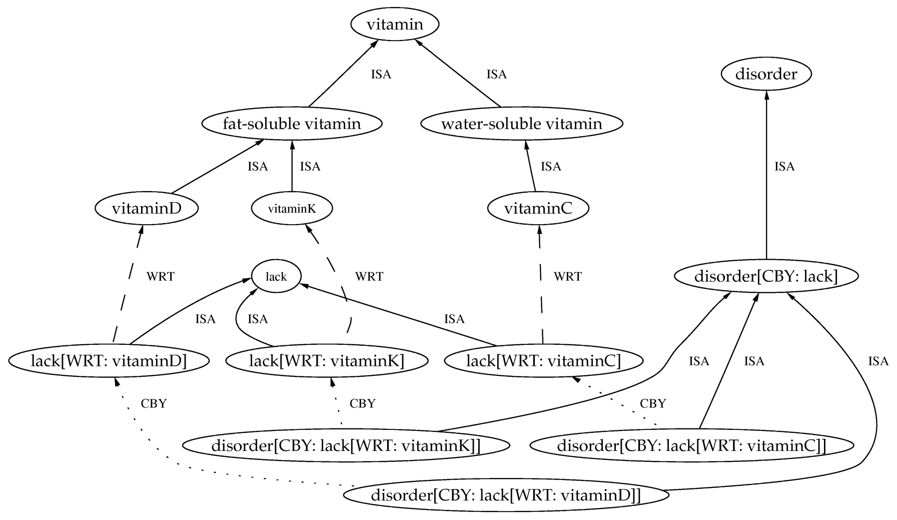


Fig. 5. A graph covering part of the ontology with semantic relations as paths.

Take as an example the ontology of figure 5. Here two semantic relations WRT and CBY are in use. The corresponding edge count functions are r^{WRT} and r^{CBY} and the attached similarity factors are denoted ρ_{WRT} and ρ_{CBY}.

Figure 6 shows the transformed graph with the attached similarity factors as weights, again assuming that the degree to which a concept Y is similar to a given concept X can be derived as the maximum of the products of edge weights over the set of paths connecting X and Y.

The attached weights are in the example assigned values in a rather ad hoc manner. Such assignment in practice needs a careful effort by domain experts. Furthermore the similarity principle in general needs to be verified empirically.

4 Conclusion

We have described three different principles for measuring similarity between both atomic and compound concepts, all of which incorporate meta knowledge.

– Similarity between atomic concepts based on distance in the ordering relation of the ontology, concept inclusion (ISA).
– Similarity between general compound concepts based on subsumption expansion.
– Similarity between general compound concepts based on distance in the ontology, but aggregated with distances in the semantic relations.

The notion of measuring similarity as distance, either in the ordering relation or in combination with the semantic relations, seems to indicate a usable theoretical foundation for design of similarity measures.

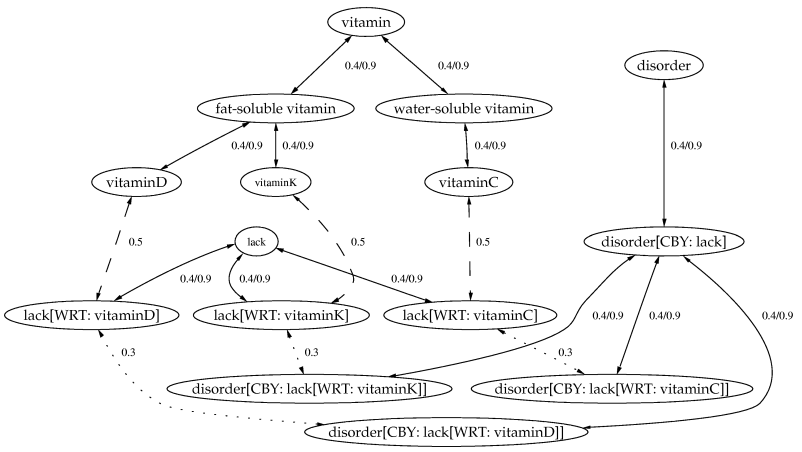


Fig. 6. The ontology of figure 5 transformed into a directional weighted graph with the similarity factors for specialization: $\sigma = 0.9$, for generalization: $\gamma = 0.4$, for CBY: $\rho_{CBY} = 0.3$ and for WRT: $\rho_{WRT} = 0.5$. The upwards and downwards edges from figure 4 describing generalization and specialization respectively, are here merged into one edge pointing in both directions. The weights are given as a pair generalization/specialization, respectively, as label on the edges. Similarity is derived as maximal (multiplicative) weighted path length, thus $sim(disorder[CBY\colon lack[WRT\colon vitaminD]], lack) = 0.3 * 0.4 = 0.12$.

The purpose of similarity measures in connection with querying is of course to look for similar rather than for exactly matching values, that is, to introduce soft rather than crisp evaluation. As indicated through examples above one approach to introduce similar values is to expand crisp values into fuzzy sets including also similar values. Expansion of this kind, applying similarity based on knowledge in the knowledge base, is a simplification replacing direct reasoning over the knowledge base during query evaluation. The graded similarity is the obvious means to make expansion a useful - by using simple threshold values for similarity the size of the answer can be fully controlled.

References

[1] Andreasen, T.: On knowledge-guided fuzzy aggregation. In *IPMU'2002, 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 1-5 July 2002, Annecy, France

[2] Andreasen, T.: Query evaluation based on domain-specific ontologies. In *NAFIPS'2001, 20th IFSA / NAFIPS International Conference Fuzziness and Soft Computing*, pp. 1844-1849, Vancouver, Canada, 2001.

[3] Andreasen, T., Nilsson, J. Fischer & Thomsen, H. Erdman: Ontology-based Querying, in Larsen, H.L. *et al.* (eds.) Flexible Query Answering Systems, *Flexible Query Answering Systems, Recent Advances*, Physica-Verlag, Springer, 2000. pp. 15-26.

[4] Andreasen, T., Jensen, P. Anker, Nilsson, J. Fischer, Paggio, P., Pedersen, B. Sandford & Thomsen, H. Erdman: *OntoQuery:* Ontology-based Querying of Texts, AAAI 2002 Spring Symposium, Stanford, California, 2002.

[5] Andreasen, T., Jensen, P. Anker, Nilsson, J. Fischer, Paggio, P., Pedersen, B. Sandford & Thomsen, H. Erdman: Ontological Extraction of Content for Text Querying, to appear in NLDB 2002, Stockholm, Sweden, 2002.

[6] Jensen, P. Anker & Skadhauge, P. (eds.): Proceedings of the First International OntoQuery Workshop *Ontology-based interpretation of NP's*, Department of Business Communication and Information Science, University of Southern Denmark, Kolding, 2001, to be republished at www.ontoquery.dk.

[7] Meadow, C.T., Boyce, B.R., Kraft, D.H.: Text information retrieval systems, second edition, Academic Press, 2000.

[8] Nilsson, B. Madsen and B. Sandford Pedersen and H. Erdman Thomsen: Semantic Relations in Content-based Querying Systems: a Research Presentation from the OntoQuery Project, in [6].

[9] Nilsson, J. Fischer: A Logico-algebraic Framework for Ontologies ONTOLOG, in [6].

[10] OntoQuery project net site: www.ontoquery.dk

[11] Yager, R.R.: On ordered weighted averaging aggregation operators in multicriteria decision making, in IEEE Transactions on Systems, Man and Cybernetics, vol 18, 1998.

[12] Yager, R.R.: A hierarchical document retrieval language, in Information Retrieval vol 3, Issue 4, Kluwer Academic Publishers pp. 357-377, 2000.

Representation & Querying of Temporal Conflict

Panagiotis Chountas[1], Ilias Petrounias[2], Krassimir Atanassov[3], Vassilis Kodogiannis[1], and Elia El-Darzi[1]

[1] Department of Computer Science, University of Westminster Watford Road, Northwick Park, London, HA1 3TP, UK
chountp@wmin.ac.uk

[2] Department of Computation, University of Manchester Institute of Science & Technology, UMIST, PO Box 88, Manchester M60 1QD, UK

[3] CLBME – Bulgarian Academy of Sciences, Acad. G. Bonchev Str., Bl, 105, Sofia-1113, BULGARIA

Abstract. In this paper we are concerned with the problem of designing of a conceptual framework for the representation needs of conflicting but still certain data, in multi-source temporal environments. We put forward a proposal based on the assumption that conflict and uncertainty are dual properties. We believe that the querying of certain but conflicting information bases will always provide possible but not always definite answers. We start by positioning conceptual schema, database with regard to this problem as part of a three-layer architecture for capturing and representing the dual properties conflict and uncertainty. Then we distinguish lattice structures as the central task in query answering considering the conceptual, instance level always with respect to the dynamism-dimension of data.

1. Introduction

The increased need for exchange of information between independent applications makes it necessary to derive information from a pool of information sources. These sources may range from relations or relational fragments within a single distributed environment or from several independent sources. In such cases intentional or extensional conflict may arise.

Intentional conflicts focus on schematic differences also referred to as schematic differences, also referred to as semantic heterogeneity. These include differences, unit differences, type differences, format or structural differences. Each such intentionally inconsistency requires an appropriate translation to convert one format from another. Once a translation is defined, the extensions are comparable.

In contradistinction, extensional inconsistencies surface only after all intentional inconsistencies have been resolved, at a point where the particular information sources may be assumed to have identical intentional representation for all overlapping information. At this point it is possible that more than one information source would provide different answers to the same user request, causing the appearance [1] of what is called "*conflict or technical uncertainty*". For example two sources may report two "different" locations as the possible location for holidays, for a particular traveller, at the same or "different" overlapping dates.

Extensional inconsistencies generate subjective uncertainty since the underlying data to be fed through the information extraction mechanism may not be uncertain however the information extraction mechanism has to make a subjective opinion on a set of facts that assumed to be definite but still contradicting against each other. In such cases the information extraction mechanism cannot define with certainty the true answer towards a particular request

This complementary problem of extensional inconsistencies or has received much less attention. Not enough attention has been paid in the case of temporal extensional inconsistencies as part of multisource information environment. The resolution of conflict indicates that uncertainty is not only a property of a particular domain or source is beyond the level of data or concepts.

2. Motivation

In this paper we deal firstly with conceptual schema conflict representation i.e. the process of encoding conflict & uncertainty in semantically rich temporal schemas. It is believed that uncertainty and conflict are dual properties. Uncertainty apart from being a property of incomplete data, or uncertain knowledge it can definitely be property of the information extraction mechanism in a multi-source environment inhabited by extensional inconsistencies either factual or temporal" It will always be an intrinsic part of multi-source environment independently of the existence of uncertain data, due to conflicting although certain data. It can be said with certainty that conflicting information will generate incomplete or partial answers to certain or complete user requests either at the conceptual or instance level.

The original motivation is based on the argument that in order to build applications with its own information bases out of the existing ones conflicting or not we need to raise the abstraction of operations on conceptual schemas and mappings.

After representing conflict at the conceptual level in terms of the structural components involved in conflict resolution, it seems appropriate to resolve conflict at the instance or information level. We propose a set of relational algebra operations that will let users to extract information from a conflicting multi-source environment. Thus we are letting users to view and query the data level from their own conceptual background, with the aid of a lattice-based environment-ontology. Technically we are permitting different domains to claim the same fact instance(s).

3. Related Issues

An intelligent query-answering system is able to perform a query, which is some indication of a user's needs, to an information base, and converts it into an answer of useful information. The focus is on extending the relational model and query language to accommodate uncertain or imprecise data with respect to value imperfection [2],

[3], [4] or temporal imperfection [5], [6], [7]. In the case of value imperfection the focus is on the representation of null values, or partial values. Alternately in temporal imperfection the focus is on the representation of indefinite temporal information. Another paradigm for query answering systems has been the logic or deductive approach, with an aim to minimize failing queries [8]. A query is said to fail whenever its evaluation produces the empty answer set.

With reference to the above post relational query answering systems, some useful observations can be deduced

– Current approaches consider Temporal or Value Imperfection as part of a single-source environment and mainly as a data dependent characteristic.
– Temporal or Value imperfection can be a property of the query answering system apart from being a data oriented characteristic. This may occur because a set of information bases or sources may provide different answers to the same query. However not all answers are assume to be equally good. At this point it quite valid to assume that all answers are perfect. Thus there may be no uncertain descriptions.

Conclusively it can be said that conflicting information sources may be responsible for queries with no answers or conflicting answers. Considering the dynamism or dimensionality of the application data and a multi-source environment a query may fail, to retrieve a unique answer for a particular fact, "*factual-conflict*" or to define through its evaluation the time-interval that a fact is defined in the real world, "*temporal-conflict*".

In this paper the authors demonstrate how conflicting and certain temporal fact instances may generate uncertainty either at the conceptual level (metadata level) or at the instance level, showing that uncertainty and conflict are dual properties.

4. Reconciling Conflicting Sources

To define a common integrated info-base, we shall assume that there exists a single (hypothetical) info-base that represents the real world. This ideal info-base includes only perfect descriptions.

We now formulate two assumptions for the governing purposes of the hypothetical info-base. These assumptions are similar to the Universal Scheme Assumption and the Universal Instance Assumption [16], although their purpose here is quite different. These two assumptions are statements of reconciliation of the given Info-bases.

- **The Scheme Soundness Principle (SSP)**. All conceptual schemes are derivatives of the real world scheme. That is, in each conceptual scheme, every structural component is a view of the real world scheme. The meaning of this assumption is that the different ways in which reality is modelled are all correct; i.e., there are no modelling errors, only modelling differences. To put it in yet a different way, all intentional inconsistencies among the independent conceptual schemes are reconcilable.
- **The Instance Soundness Principle (ISP)**. All database-fact instances are derivatives of the real world instance. The meaning of this assumption is that the information stored in info-bases is always correct; i.e., There is no erroneous information, only different semantic representations of alike facts.

It is suggested by [9] that in query-answering systems one central task is to compare two information items, one from the client and the other from the info-base. In our framework a client can be defined as

– An application that is trying to built its own info-base out of the existing reconcile able info-bases according to **Scheme Soundness (SSP)**, and **Instance Soundness (ISP)**, principles
– An unaware hypothetical user trying to extract information similar to its request from the info-base, that is the closest one to the single (hypothetical-perfect) database which represents the real world according to **(SSP)**, **(ISP)** principles.

The question to be answered at this point is how [9] argument is realized in a multi-source, reconcile-able although conflicting environment?

At this point we will introduce first a rich semantically temporal conceptual model that utilized by the single (hypothetical-ideal) info-base to represents the real temporal world. Afterwards we will propose an evidential model for resolving conflict at the instance level through the query mechanism.

5. The Meta-model for Capturing Uncertainty & Conflict

The meta-model for the proposed formalism specifies the basic concepts of the conceptual model that are involved in uncertainty-conflict reasoning and is application independent.

Time facts or temporal phenomena are presented as objects in a 3 dimensional space spanned by the time axis, the uncertainty axis, and the axis along which facts. A fact is a true logical proposition about the modelled world. The building blocks are facts in which entity types play roles.

$$F = \{\{<E_1, L_1, R_1> \ldots <E_n, L_n, R_n>\}, \{\Delta t \leq T\}\}$$

Where:
F is a fact type consisting of k fact instances
T is the time interval that an irreducible fact type is defined in the real world.
E_i is the entity type playing a role in the fact type
L_i is the label type (referencing E_i)
R_i is the role of the fact type
Δt is the time interval that an irreducible fact instance is defined in the real world

Fig. 1. Fact Type

Each entity type is a set of elements and contains all concepts regardless of the different roles that they play in any of the different facts. A role played by an entity type in a fact defines a subset of an entity type. A fact type F_x among n entity types (E_1, E_2,…,E_n) is a set of associations among elements from these types. A label is used to reference an entity type and refers to elements from the domain of an entity type.

Factual uncertainty is related to possible semantic propositions. This leads to possible associations between entity instances and not between entity types. Factual uncertainty is the result of ambiguity at the instance level. In our approach there is no ambiguity at the level of concepts.

An irreducible fact type is defined in the real world over a time interval or time period (valid time). This is called a timestamped fact. Temporal uncertainty in the information context specifies that a semantic proposition [10] in the real world be defined over a time interval with no explicit duration.

Comparing the definition of factual uncertainty and temporal uncertainty, it can be concluded that these two forms of uncertainty are independent. Certain fact propositions may be defined over an uncertain time, or uncertain fact instances may exist during a certain time and vice versa. Timestamped facts are defined as cognisable since uncertainty about time and fact are independent. If a time fact is cognisable either the time fact or the independent time and fact representations are sufficient to represent all knowledge, provided that correct 2-dimensional representations can be derived from the time fact (3-dimensional space) and not the other way around. In this case the range of possible time points which correspond to the earliest possible instantiation F_s of the fact F_x is constrained, as well as the range of time corresponding to the latest instantiation F_e. Similarly the range of the possible instantiations of a fact, corresponding to the earliest time point instantiation t_s of F_x is constrained, as also those fact instantiations corresponding to the latest time t_e. These constraints are summarised in equation (1).

$$F_s \rightarrow [t_s, t_1] \qquad F_e \rightarrow [t_2, t_e]\ , \quad t_s \rightarrow [F_s, F_1] \quad t_e \rightarrow [F_2, F_e] \tag{1}$$

In equation (1) t_s, t_1, t_2, t_e are linear points and as constraints are applied to the time fact instantiations. A temporal phenomenon is defined over the union of the time intervals specified by the temporal constraints in equation (1). Therefore a temporal phenomenon is defined over a temporal element. This is described in equation (2).

$$F_x = \{F_s,\ldots,F_e\} \rightarrow \{[t_s, t_1],.., [t_2, t_e]\} \tag{2}$$

All the above axioms and concepts are represented in the uncertainty metamodel Fig.1. The meta-model also describes the different types of uncertainty and the different temporal phenomena. The latter might be unique, recurring or periodic.

Conflicting information may generate two types of uncertainty. One is introduced because of queries that refer to level concepts that are at a lower level than those that exist in the instance level of the database. The other arises because of the use of an element in the query that is a member of more than one high level concept. In terms of dimensions uncertainty that is generated because of conflicting information may be either factual or temporal.

If the duration of a fact can be expressed with the aid of more than one time granularities then "*technical temporal conflict*", Fig.3.b, is defined and temporal uncertainty is generated, since the answering query mechanism is not in position to clarify which time granularity is the most preferred one- the one that totally satisfies the temporal selection condition.

If a reference label is claimed by more than high-level concepts the "*factual conflict* ", Fig.3.a, is defined and factual uncertainty is generated, since the answering query mechanism is not in position to clarify which reference label is the most preferred one -the one that totally satisfies the factual selection condition.
The rest of the paper is attempting to address integration issues that may come up, through query resolution in a flexible manner.

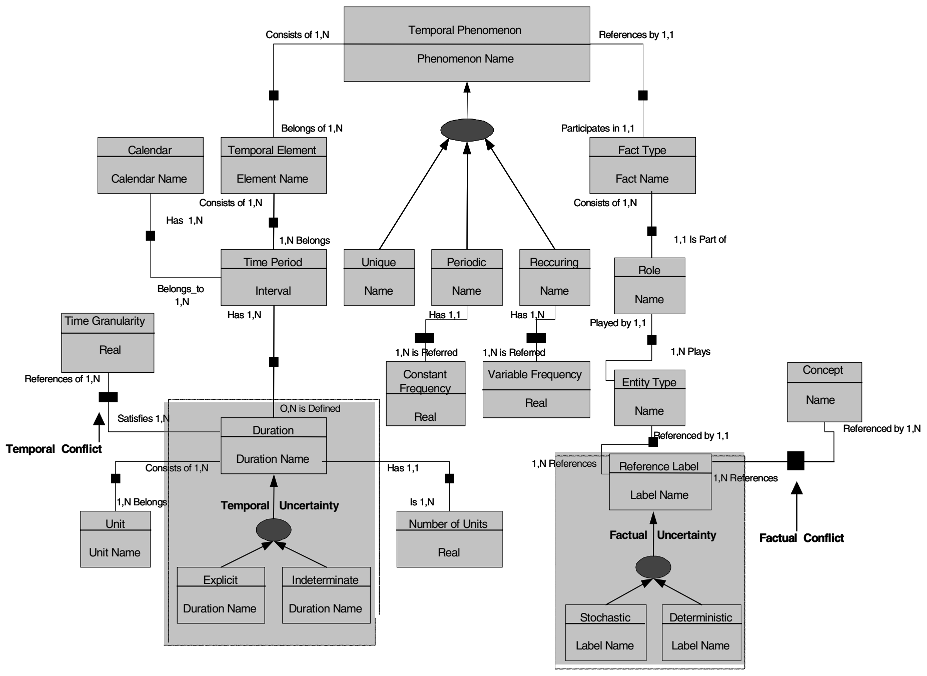


Fig. 2. Uncertainty & Conflict Meta-model

6. Problems in Estimating Flexible Answers

Let us consider the following description about travellers 'Ann' & 'Liz' ", Table I.
Consider the following conflicting queries and whether these could be answered with authority or not:

– Q_1: When either did Liz or Ann visit Brazil?
– Q_2: Derive all the people who either did visit the Southern Hemisphere or the Northern Hemisphere?

Table 1. Relation R representing the schedules of Ann and Liz

R	Person	Concept	VT(R)
X_1	Ann	Brazil	[01/03/00,29/05/00]
X_2	Ann	Southern Hemisphere	[01/05/00,29/08/00]
X_3	Ann	Northern Hemisphere	{[01/06/00,29/08/00], [01/09/00,29/11/00]}
X_4	Liz	Brazil	[01/05/00,29/08/00]
X_5	Liz	Northern Hemisphere	{[01/06/00,29/08/00], [01/09/00,29/11/00]}

The above queries are not easy to be answered because of the following inconsistencies: If we consider a lattice-structured domain, then Brazil has two parents (Northern Hemisphere and Southern Hemisphere, as shown in Fig.3.a). Therefore Q_2 is not easy to be answered. We cannot estimate with precision the exact date of arrival and stay in Brazil for both travellers (Liz, Ann as shown in Fig.3.b). Therefore Q_1 is not easy to be answered either. The above inconsistencies are presented in Table I through relation R. Another useful observation is that compound selection statements are useful in expressing the separable notions of factual and temporal conflict.

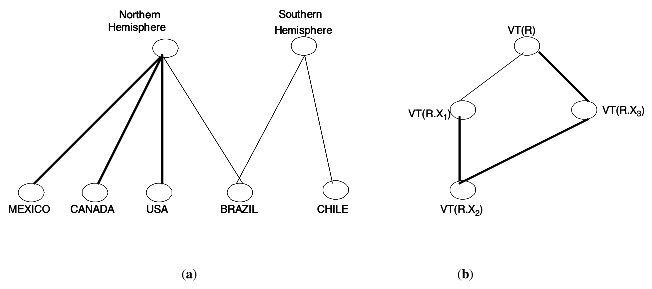


Fig. 3. (**a**) Lattice of Concepts and (**b**) Lattice of time intervals

An analytical observation of Table I is rising the following requirements/issues in terms of data representation:

A time model and representation that presents temporal information in terms of the following physical measurements duration *D* (e.g. tuples {$\mathbf{X_1, X_2, X_4}$}) and frequency K of reappearance, if information is periodical, where *D*, *K* may not be known, (e.g. tuples {$\mathbf{X_3, X_5}$}).

In our effort to classify tuples in Table I based on a virtual decision attribute which simply declares the fact that a person has visited Brazil, it can be seen that tuples {X_2, X_3, X_5} cannot be classified with an exclusive Boolean Yes or No. A similar problem would arise if a user requested through an algebraic operation all the countries that Ann or Liz did visited in the Northern and Southern Hemispheres. Therefore, there is also a need for algebraic operations that will support approximate answers.

7. The Time Representation

The time representation has the flexible feature [5], [6], [11] of representing simultaneously the three different types of temporal information (definite, indefinite, infinite).

A time interval Δt is presented in the form of $[C+K*X,\ C'+K*X]$ where $C'=C+D$, $D\in N^*$,(*D denotes the property of duration*) thus an interval is described as a set of two linear equations defined and mapped over a linear time hierarchy (e.g. H_2 = day$\subseteq$month$\subseteq$year). The lower time point $t_{Earliest}$ is described by the equation $t_{Earliest} = C+K*X$. The upper point t_{Latest} is described by the equation $t_{Latest} = C'+K*X$. C is the time point related to an instantaneous event that triggered a fact, K is the repetition factor, $K\in N^*$ or the lexical 'every' (infinite-periodical information). X is a random variable, $X\in N$, including zero, corresponding to the first occurrence of a temporal fact (phenomenon) restricted between 0 and n.

$$t_L = t_{Earliest} = C+K*X,\ t_R = t_{Latest} = C'+K*X,\ \ C' = C+D,\ 0\leq X\leq n,\ x\in N^*,\ \ K>0 \qquad (\mathbf{3})$$

The above time formalism will enable an application or a user to express the following types of temporal information

Definite Temporal Information: The duration (D) of an event is constant (D = t). All times associated with facts are known precisely in the desired level of granularity, assuming that K=0, since a definite fact occurs only once.

Indefinite Temporal Information: is defined when the time associated with a fact has not been fully specified. Therefore the duration of a fact is indeterminate or *bounded* ($C_L \leq D \leq D_R$).

When $0\leq X\leq n$, $x\in N^*$ and $K>0$, then *Infinite Temporal Information* is represented. *Infinite Temporal Information* is defined when an infinite number of times are associated with a fact. Infinite temporal information includes the following types of information:

– *Periodic*: An activity instance is repeated over a time hierarchy with the following characteristics: a constant frequency of repetition K, it has an absolute and constant duration D, and X a random variable that denotes the number of reappearance's for an activity instance.
– *Unknown Periodic*: Generally is described in the following intervalic form, $\Delta t = [\perp, \perp]$. The intuition is that the duration D of an activity instance is assumed to be known. However the frequency of reoccurrence ($K=?$) is not known. Recalling the definition it is known that if an activity instance is periodical, then its next reappearance cannot occur before the previous one is ended. It is known that $K \geq 1$. Therefore the following conclusion can be made $1\leq D\leq K$.

The product $K*X$ is mapped also to a linear time hierarchy (e.g. H_2 = day$\subseteq$month$\subseteq$year). D represents the duration of a timestaped fact instance, is also mapped in a linear time hierarchy and may be in the range between a lower and upper bound $G_L \leq D \leq G_R$. Constraints are built from arbitrary linear inequalities.

8. Representation of Conflicting & Uncertain Information

Let T be a set of time intervals $T=\{[t_L, t_R]$ where $t_L=C+K\times X$, $t_R=C+K\times X \wedge a_l\leq X\leq a_v\}$ and D a set of non temporal values. A generalised tuple of temporal arity x and data arity l is an element of $T^x \times D^l$ together with constraints on the temporal elements. In that sense a tuple can be viewed as defining a potentially infinite set of tuples. Each extended relation consists of generalised tuples as defined above. Each extended relation has a virtual tuple membership attribute formed by a selection predicate either value or temporal that models the necessary (*Bel*) and possible degrees (*Pls*) to which a fact-instance belongs to the relation. The domain of tuple membership attribute is the Boolean set Ω = {true, false}. The possible subsets to that are {true}, {false} and Ω. The support set for tuple membership can be denoted by a pair of numbers: (*Bel*, *Pls*) where: *Bel* = m {|true|},*Pls* = m{|true|} + m{Ω} with property $0\leq Bel\leq Pls\leq 1$

A tuple with (*Bel*, *Pls*) = [1,1] corresponds to a tuple that qualifies with full certainty. A tuple with (*Bel*, *Pls*) = [0,0] corresponds to a tuple that is believed not to qualify with full certainty. A tuple with (*Bel*, *Pls*) = (0,1) corresponds to complete ignorance about the tuple's membership. At this point an issue arises: the estimation of the (*Bel*, *Pls*) measures in a structured lattice domain

Let l be an element defined by a structured domain L. $U(e)$ is the set of higher level concepts, i.e. $U(e) = \{n|n \in L \wedge n$ is an ancestor of $l\}$, and $L(e)$ is the set of lower concepts $L(e) = \{n|n \in L \wedge n$ is a descendent of $l\}$. If l is a base concept then $L(e) = \emptyset$ and if l is a top level concept, then $U(e) = \emptyset$. If L is an unstructured domain then $L(e) = U(e) = \emptyset$. Considering tuple $\{X_2, X_3, X_6\}$ and the selection predicate *location = "Brazil"* then $L(e)$, $U(e)$are defined as follows:

- U(Brazil) = {Southern Hemisphere, Northern Hemisphere} L(Brazil) = $\emptyset$

Rule 1: If ($|U(e)| > 1 \wedge L(e) = \emptyset$), e.g. $|U$(Brazil)$|$=2, L(Brazil) = $\emptyset$, then it is simply declared that a child or base concept has many parents (lattice structure). Therefore a child or base concept acting as a selection predicate can claim any tuple (parent) containing elements found in $U(e)$, as its ancestor, but not with full certainty ($Bel > 0$, $Pls \leq 1$). This is presented by the following interval (*Bel*, *Pls*) = (0,1].

Now consider the case where the selection predicate is defined as follows *location = "Southern Hemisphere"*. Tuple $\{X_2\}$ fully satisfies the selection predicate and thus (*Bel*, *Pls*) = [1,1]. Tuples $\{X_3, X_5\}$ do not qualify as an answer and thus (*Bel*, *Pls*) = [0,0]. However it cannot be said with full certainty ((*Bel*, *Pls*) = [1,1]) whether $\{X_1, X_4\}$ satisfy the selection predicate or not, since Brazil belongs to both concepts {Southern Hemisphere, Northern Hemisphere}. Using the functions $L(e)$, $U(e)$ this can be deduced as follows

- U(Southern Hemisphere) = {$\emptyset$}, L(Southern Hemisphere) = {Brazil, Chile}
- $B=U(L$(Southern Hemisphere)$\wedge$(R.*concept*)) = U(Brazil) = {Southern Hemisphere, Northern Hemisphere}. Formally the function can be defined as follows:

$B((l_1),(l_2))= U(L(l_1)\wedge(l_2))$ where l_1 is a high level concept, l_2 is a base concept are elements defined in a lattice structured domain. If both arguments are high level concepts or low level concepts then $B((l_1),(l_2))= \emptyset$. Function $B((l_1),(l_2))$ is defined only in a lattice structured domain.

Rule 2: If $B((l_1),(l_2))$ is defined and $|B((l_1),(l_2))|>1$, then it is simply declared that multiple parents, high level concepts, are receiving a base concept as their own child.

Therefore a parent or high level concept acting as a selection predicate can claim any tuple (child) containing elements found in ($L(l_1)\wedge(l_2)$), as its descendant, but not with full certainty ($Bel > 0$, $Pls \leq 1$), presented by the following interval (Bel, Pls) = (0,1].

Similarly a temporal selection predicate can use the above functions ($U(e)$, $L(e)$, $B((l_1),(l_2))$) for imprecise temporal information, representing the time dimension as intervals, by labelling each node in the lattice with a time interval. The use of a lattice-structured domain by an application permits also the representation of temporal information at different levels of granularity.

Next an extended relational algebra is defined which operates on our model. The operations differ from the traditional ones in several ways: The selection/join condition of the operations may consist of base concepts or high-level concepts.

9. Representation of Conflicting & Uncertain Information

We are considering, for illustration purposes, the three operations σ (selection), $\bowtie$ (join), $\cup$ (set union).

Selection: Selection is defined as follows: $\sigma_P(R) := \{t \mid t \in R \wedge P(t) = \text{true}\}$ where P denotes a selection condition. There are two types of a selection condition. A data selection condition (P_d) considers the snapshot relation R in Table 1. The temporal selection condition (P_t) is specified as a function of three arguments $P_t: = < K,D,C>$ which is mapped to the time hierarchy H_r. It has to be mentioned that temporal constraints are included in the result tuples. The combined predicate over relation VT(R) in Table II is defined as follows: $P:= P_d \mid P_t \mid P_d \wedge P_t$. The selection support function $F_s(t_{A1..An}, P)$ returns a (Bel, Pls) pair indicating the support level of tuple t for the selection condition P, where $A_1..A_n$ is the set of attributes, excluding the virtual membership attribute. The selection support function F_s utilises the ($U(e)$, $L(e)$, $B((l_1),(l_2))$) functions in conjunction with Rule-1 and Rule-2, as defined in section 7, for estimating the actual support values. Recall that a compound predicate is formed by a conjunction of two or more atomic predicates. In this paper it is assumed that the atomic predicates are mutually independent. The support for the compound predicate $P:= P_d \mid P_t \mid P_d \wedge P_t$ is computed based on the multiplicative rule:

$$F_s(P) = (F_s(t_{A1..An}, P_t) \wedge F_s(t_{A1..An}, P_d)) = (Bel_1 \times Bel_{2.....} \times Bel_n,\ Pls_1 \times Pls_2 \times Pls_n). \qquad \textbf{(4)}$$

Projection: $\pi_X(R) := \{t(X) \mid t \in R\}$, where R is a relation on scheme S, t is a tuple with scheme X and X is a subset of S ($X \subseteq S$). Projection retains all valid time values like standard projection. Projection is defined on top of a selection. The intuition is that, as an operator it does not modify F_s, that is the tuple membership

Join: Let R, S be two extended relations, P be the join condition and Q the membership threshold condition. The extended join operator is defined as a Cartesian Product, followed by an extended selection: $R \bowtie_P^Q S \equiv \sigma_P^Q(R \times S)$ where the tuple membership function is deviated by F_s (1) as in the case of the extended select operation. The time interval that the tuple membership is defined over is the intersection of the time intervals that the sources ($A_1 \ldots A_n$) are defined. $\Delta t(_{Fs})$ =

$\Delta t_1 A_1 \cap \Delta t_2 A_2 \cap_{.....} \cap \Delta t_n A_n$. Assuming two intervals with lower bounds $t_L = C + KX$ and $t_{L'} = C' + K_1 X$ and upper bounds $t_R = C_1 + KX$ and $t_{R'} = C_1' + K_1 X'$. The time interval for the result is defined as $t_{L'} = C'' + K'X$ the common lower bound where $C' = \max(C, C')$, and $K' = \min(K, K_1)$, and $t_{U'} = C_3 + K'X$ the common upper bound where $C_3 = \max(C_1\ C_1')$, and $K' = \min(K, K_1)$.

Union: For the set operators, including union, uncertainty can be introduced when relations with different levels of refinement for the same information are combined. Without extra knowledge it is reasonable to choose the information with the finest granularity as the one to be classified with full certainty (*Bel*, *Pls*) = [1,1]. Information not in the finest granularity is classified with no full certainty (*Bel*, *Pls*) = (0,1]. Both types of information are part of the result tuples, accompanied by different beliefs. Union is formally defined as follows:

$R \cup S \equiv \{t | (\exists r)(\exists s)(r \in R \wedge s \in S \wedge t.L = r.L = s.K) \wedge (t(Bel, Pls) = F_s(r.(Bel, Pls), s(Bel, Pls))$. L is the arity of the relation, F_s denotes the selection support function. Tuples with different valid times are not merged, independently of the fact that they are expressing the same snapshot tuple.

10. Conclusions

The need to build applications with its own information bases out of the existing ones conflicting or not we need to raise the abstraction of operations on conceptual schemas (i.e. ER, ORM) and mappings and its temporal extensions. The intuition is that the integrated temporal conceptual schema (S_{AB}) must express equally well the similarities and dissimilarities (conflict), between schemas (S_A, S_B) under integration. In this sense similarity functions that measure the conceptual match m_c and the relational similarity m_r have to be defined as part of an equivalent graph formalism [ ] that corresponds to conceptual schema (S_{AB}). The conceptual match m_c express how many concepts the two schemas (S_A, S_B) have in common. The relational similarity m_r indicates how similar the relationships between the same concepts in both facts. In away it shows how similar the contexts of the common concepts in both facts are. However it has to be emphasised that conflict resolution at the conceptual level will only resolve intentional conflict. Extensional conflict resolution will be a separate issue since it is related to data domains.

Further on we are outlying some thoughts that will enable us to represent conflicting information as part of a 4 valued characteristic function of paraconsistent relation, which maps tuples to one of the following values: $\top$ (for contradiction), t (for true), f (for false) and $\perp$ (for unknown). This will let to reason about conflict a quantitative four-value logic. The elements of the temporal information (temporal facts) can be represented in the form $< F, t_L^F, t_R^F >$, where $[t_L^F, t_R^F]$ is a time interval [12]. Using the ideas for intuitionistic fuzzy expert systems [12], we can estimate any fact F and it can obtain intuitionistic fuzzy truth-values $V(F) = < \mu^F, \nu^F >$, such that $\mu^F, \nu^F \in [0,1]$ and $\mu^F + \nu^F \leq 1$. Therefore, the above fact can be represented in the form $< F, t_L^F, t_R^F, \mu_F, \nu_F >$. This form of the fact corresponds to the case in which the fact is valid in interval $t^F = [t_L^F, t_R^F]$ and at every moment of that interval the fact has the truth-value $<\mu^F, \nu^F>$. Thus it is required to develop a relational intuitionistic environment for representing uncertainty and contradiction or conflict as part of a multi-source temporal database environment.

References

1. Motro A, Anokhin P, Berlin J, Intelligent Methods in Virtual Databases, Proceedings of the Fourth International Conference on Flexible Query Answering Systems, FQAS 2000, Advances in Soft Computing, *ISBN 3-7908-1347-8,* Physica-Verlag, 580-590, 2000
2. Barbará, D., Garcia-Molina H., Porter, D., The Management of Probabilistic Data, IEEE Transactions on Knowledge and Data Engineering, Vol. 4, No.5, 487-502, 1992
3. Fuhr N., Rölleke T. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems, ACM Transactions on Information Systems, TOIS Vol. 15, No 1, pp 32-66, 1997
4. Adnan Yazici, Alper Soysal, Bill P. Buckles, Frederick E. Petry: Uncertainty in a Nested Relational Database Model. DKE 30(3), 275-301, 1999
5. Chountas, P, Petrounias, I., Representation of Definite, Indefinite and Infinite Temporal Information, Proceedings of the 4th International Database Engineering & Applications Symposium (IDEAS'2000), *IEEE Computer Society, ISBN 0-7695-0789-1*, 167-178, 2000
6. M. Koubarakis, Representation and Querying in Temporal Databases: the Power of Temporal Constraints, Proc.of the 9th International Conference on Data Engineering (ICDE), 327-334, 1993
7. Dyreson, C.E., Snodgrass. R.T., Support Valid-Time Indeterminacy, ACM Transactions on Database Systems, Vol. 23, No. 1, 1-57, 1998
8. T. Bylander, T. Steel, The logic of Questions and Answers, Yale University Press, New Haven, CT, 1976.
9. Andreasen T, Christiansen H, Larsen H. L, *Flexible Query Answering Systems,* Kluwer Academic Publishers, 1997
10. Chountas, P, Petrounias, I., Representing Temporal and Factual Uncertainty, Proceedings of the Fourth International Conference on Flexible Query Answering Systems, FQAS 2000, Advances in Soft Computing, *ISBN 3-7908-1347-8,* Physica-Verlag, 161-170, 2000.
11. F. Kabanza, J-M. Stevenne, P. Wolper, Handling Infinite Temporal Data, Proc. of ACM Symposium on Principles of Database Systems (PODS), 392-403, 1990
12. Atanassov K., Temporal Intuitionistic fuzzy relations. Proceedings of the Fourth International Conference on Flexible Query Answering Systems, FQAS 2000, Advances in Soft Computing, *ISBN 3-7908-1347-8,* Physica-Verlag, 153-160, 2000.

Adding Flexibility to Structure Similarity Queries on XML Data

Paolo Ciaccia and Wilma Penzo

DEIS - CSITE-CNR
University of Bologna, Italy
{pciaccia,wpenzo}@deis.unibo.it

Abstract. The presence of structure inside XML documents poses the hard challenge of providing flexible query matching methods for effective retrieval of results. In this paper we present an approach that faces this issue in a twofold fashion: 1) it exploits new approximations on data structure; 2) it provides a relevance ranking method that takes into account the degree of *correctness* and *completeness* of results with respect to a given query, as well as the degree of *cohesion* of data retrieved.

1 Introduction and Motivation

The increasing availability of large digital libraries is raising XML to be the new standard for data representation. Because of the heterogeneity of document collections, querying blindly XML data requires a great deal of flexibility not to fall into the trap of "empty results", often caused by too strict constraints. In fact, the presence of structure inside XML documents is double-faced: on one hand, it helps to define the context where information has to be retrieved (for instance, "Retrieve the director of the movie entitled *Casablanca*"), on the other, it may be an obstacle to the retrieval of relevant information that slightly differs in data organization.

For instance, consider a user looking for stores in New York city selling CD's authored by Elton John, and containing songs with "love" in the title. Among the two stores listed in `Doc1` of Fig. 1, only the former fully satisfies query requirements. This is because it is the only store in `Doc1` that presents a `city` attribute, thus making condition on "New York" checkable. But, it is evident that also the second store is relevant to the query, and should be returned.

Thus, relaxation on query requirements is currently the scope of many proposals [4,7,10,19,20,21]. These works deal with similarity queries that approximate results both on semantics and on structural conditions. However, most of the above approaches do not recognize the second CD store of `Doc1` as relevant. Actually, except for ApproXQL [17], they do not cope with the problem of providing *structurally incomplete results*, i.e. results that only partially satisfy query requirements on data organization. Further, although supposing that condition on New York is checkable for the second CD store in `Doc1`, none of the above

```
<cdstore>Artist Shop
  <address street = "105 Broad Street",
        city = "New York" >
  <cd>
    <title>One night only</title>
    <artist>Elton John</artist>
    <tracklist>
      <song>
        <title>
          Can you feel the love tonight
        </title>
      </song>
      ...
    </tracklist>
  </cd>
  ...
</cdstore>
<cdstore address ="Manhattan, New York" >
  Music Store
  <cd>
    <title> Disney solos for violin </title>
    <tracklist>
      <track>
        <author> Elton John </author>
        <title>
          Love of the common people
        </title>
      </track>
      ...
    </tracklist>
  </cd>
  ...
</cdstore>
```

Doc1

```
<musicstore name="CD Universe">
  <location>
    <town>New York</town>
  </location>
  <warehouse>
    <stock number ="157">
      <cd>
        <title>Love songs</title>
        <singer>Elton John</singer>
        <tracklist>
          <song>
            <title>
              Can you feel the love tonight
            </title>
          </song>
          ...
        </tracklist>
      </cd>
      <cd>
        <title>
          Songs from the west coast
        </title>
        <singer>Elton John</singer>
        <tracklist>
          <song>
            <title>I want love </title>
          </song>
          ...
        </tracklist>
      </cd>
    </stock>
  </warehouse>
</musicstore>
```

Doc2

Fig. 1. Sample XML documents

method would recognize it as relevant, because of a slight difference on data organization. In fact, Elton John appears as a track author rather than as the CD author, as specified by the query. This emphasizes the need of more flexibility in the evaluation of structure conditions, most of all in absence of knowledge of data organization.

As to relevance ranking, relaxations on both semantics and structure are expected to affect the score of results. However, in many proposals [4,7,10,20] the use of wildcards flatten the score of the retrieved data, which is considered relevant no matter how much "sparse" it is. Only in ApproXQL the number of elements matching a wildcard contributes to the ranking of results. But none of the above approaches takes into account neither the query *rate* satisfied by an answer, i.e. the *completeness* of the answer, as well as its *structural correctness*, nor the *cohesion* of results. In fact, also *sparseness* of data retrieved plays an important role for relevance evaluation. For instance, consider the CD store in Doc2. Although it satisfies the query conditions, note that the required CD's belong to the stock "157" of the store's warehouse. This changes the context where information is retrieved and, in this case, this may possibly mean that the

CD's are not currently available for sale. Then, this store contains two relevant CD's, thus probably it is expected to score higher than others.

In this paper we provide a method to find the *approximate embeddings* of a user query in XML document collections. Our proposal captures the relaxations described above, and provides a ranking measure that takes into account the degree of correctness and completeness of results with respect to a given query, as well as the degree of cohesion of data retrieved. Our proposal relies on tree representation both for XML documents [3], and for queries. The outline is as follows: In Section 2 we discuss related work. In Section 3 we start from the unordered tree inclusion problem [12] to relate query and data trees through *embeddings*; this is extended in two directions: 1) to capture also partial matching on query structure, and 2) to assign a score to the retrieved embeddings. In Section 4 we define the *SATES* (*Scored Approximate Tree Embedding Set*) function, to retrieve and score embeddings, and we give a flavour of the flexibility and the effectiveness of the method through examples. However, we report its formal definition in appendix. Section 5 is devoted to relevance ranking computation. Properties of correctness and completeness of embeddings are presented. We also compare our method with other related approaches. In Section 6 we briefly discuss implementation and, finally, in Section 7 we conclude and discuss our planned future work.

2 Related Work

Several current proposals [4,6,7,8,10,11,16,17,20,21] deal with similarity queries on XML data. In XXL [20] similarity basically depends on document content, through the evaluation of vague predicates, and information on structure is only exploited to combine semantic similarity scores. Partial match on query structure is not supported. XIRQL [10] introduces the concept of *index object* to specify document units. Terms are weighted locally to these units, thus defining the relevance of each term in the scope given by the node. Thus, structural conditions act as filters to determine the context where information has to be searched in. Partial match on query structure yet is not discussed. The latter feature is neglected also in [7]. The ranking of scores depends on the context where data is retrieved, and semantic similarity is not exploited.

Then, other approaches [4,17] adopt tree-based retrieval models to captures XML data relationships in a natural way. In fact, classical models (e.g. vector space, probabilistic) lack the ability to handle structural information, although some extensions have been proposed for them [18,21]. In [4] relevance ranking is based on tree pattern relaxations. Nodes and edges in a query tree are assigned pairs of weights that denote scores for exact and relaxed matching, respectively. Sum is used to combine scores of each single node/edge matching. Partial match on query structure is not supported. Only ApproXQL [17] considers partial match on query structure.

Then, in all above-cited proposals cohesion of data retrieved is not considered for relevance. In [7] the length of paths connecting two matching nodes is

neglected; In [20] wildcards are intentionally used, thus discarding the (negative) contribution of intermediate nodes between matching data; In [10] this issue is not discussed. In [17] similarity scores depend on the costs of basic transformations applied on the query tree. These costs do not depend on data. Thus, semantic relationships, like synonymy, are not exploited.
Further, to our knowledge, for a given document, all proposed methods return only the *best* (in some cases, approximate) matching. The set-oriented approach used in the $SATES$ function also captures the relevance given by multiple occurrences of the query pattern in the retrieved data.

Other related approaches deal with matching elements of data schemas, usually represented as graphs [14,15]. In [14] trees are used as a starting point to perform schema matching (for instance, to compare XML documents). Structural similarity is measured as the fraction of leaves appearing in the matching. The notion of completeness is then limited to leaf coverage. Cohesion is not explicitly discussed, although it influences the final similarity of trees, through the use of descreasing factors. In [15] quality of results is measured in terms of the effort the user needs to spend to obtain a perfect match from the automatically generated one. This is accomplished through the addition/deletion of node pairs in the mapping. This suggests a measure of incompleteness of results (when nodes are to be added), but it does not provide any information on cohesion of data retrieved.

3 Relaxing the Tree Embedding Problem

In a tree view of documents and queries, several proposals [4,7,10,17,20] rely on tree/graph pattern matching techniques to decide if a document d is a possible answer to a query q.

In most of these works, *all* query nodes must have a corresponding matching node in the document tree, and *each* parent-child relationship should be guaranteed at least by an ancestor-descendant one in the data tree. This is also known as the *tree embedding problem* [12]. Nevertheless, in many cases these conditions are too restrictive, yet being not flexible enough to effectively deal with the structural heterogeneity of XML documents. In fact, in absence of knowledge of data organization, documents often do not correspond *completely* to query requirements, yet being mostly relevant although in absence of some conditions. This is the case, for instance of the second CD store of `Doc1`, that does not present any `city` element/attribute to be checked for the query condition on New York. Further, frequently, documents do not *exactly* fit the structural constraints provided in a user query. An example is `Doc1`, where Elton John appears as track author, instead of CD author as required in the sample query presented above.

Our work basically loosens the strictness of this approach, that often leads to empty results because of minor differences in data organization. The key aspects of our proposal are:

1. the relaxation on the concept of *total* embedding of a query tree t_q in a document tree t_d, in that we admit *partial* structural match of the query tree, as well as *approximations on structural relationships*; further, the match is relaxed to consider semantic similarity between nodes;
2. *ranking* of results with respect to the *cohesion* of data retrieved, to the *relaxation* of semantic and structural constraints, and to the *coverage rate* of the query;
3. a set-oriented approach to results, that depending on different relaxations on requirements, identifies possible alternatives that the user may be interested in. This also strengthens the relevance of data presenting multiple occurrences of query patterns.

Point **1)** leads to the introduction of our interpretation of *approximate tree embedding*. First, we report the basic notation we will use throughout paper and appendix. As to tree representation of XML documents, we follow the XML Information Set standard [3] (with *root* and *label* as `document element` and `local name` properties, respectively, and *parent* and *children* as the homonymous properties, denoting a parent-child relationship between argument nodes, and the set of first level children of a given node, respectively). We also define some additional functions for nodes n, n_1, and n_2 in a tree: *leaf(n)* iff $\mathit{children}(n) = \emptyset$, and $\mathit{ancestor}(n_1,n_2)$ iff $(\mathit{parent}(n_1,n_2) \vee \exists\ n \in N$ s.t. $(\mathit{parent}(n_1,n) \wedge \mathit{ancestor}(n,n_2))$. We denote with $\mathcal{T}$ the set of such trees. Given a tree $t \in \mathcal{T}$, $\mathit{nodes}(t)$ returns the set of all nodes of t. Then, given a node n in a tree $t \in \mathcal{T}$, we define *supp(n)* the subtree of t rooted at n, and we call it the *support* of n in t. Let $\mathcal{T}_D \subset \mathcal{T}$ be the set of all data trees, and $\mathcal{T}_Q \subset \mathcal{T}$ be the set of all query trees.

We intentionally neglect the presence of namespaces and links (IDREFs) inside XML documents and we assume data is well-formed, but we do not require validity with respect to a Document Type Definition (DTD).

Definition 1 (Approximate Tree Embedding (ATE)). Given a query tree $t_q \in \mathcal{T}_Q$ and a document tree $t_d \in \mathcal{T}_D$, an *approximate tree embedding* of t_q in t_d is a *partial* injective function $\tilde{e}[t_q, t_d] : \mathit{nodes}(t_q) \nrightarrow \mathit{nodes}(t_d)$ such that $\forall q_i, q_j$ in the domain of $\tilde{e}$ ($\mathit{dom}(\tilde{e})$):

a). $\mathit{sim}(\mathit{label}(q_i),\mathit{label}(\tilde{e}(q_i))) > 0$, where *sim* is a similarity operator that returns a score normalized in [0,1] stating the semantic similarity between the two given labels
b). $\mathit{parent}(q_i,q_j) \Rightarrow (\mathit{ancestor}(\tilde{e}(q_i),\tilde{e}(q_j)) \vee \mathit{sibling}(\tilde{e}(q_i),\tilde{e}(q_j)))$
c). $\mathit{sibling}(q_i,q_j) \Rightarrow (\mathit{sibling}(\tilde{e}(q_i),\tilde{e}(q_j)) \vee \mathit{ancestor}(\tilde{e}(q_i),\tilde{e}(q_j)))$

Let $\mathcal{E}$ be the set of approximate tree embeddings.

Points *b)* and *c)* of Def. 1 capture both the presence of intermediate nodes between $\tilde{e}(q_i)$ and $\tilde{e}(q_j)$ (ancestorship), and additional approximations on structural conditions (sibling relationship), that we call *structure unbalances*, as shown in the following example:

Example 1. Recall the sample query presented in the Introduction: *Retrieve stores in New York city selling CD's authored by Elton John, and containing*

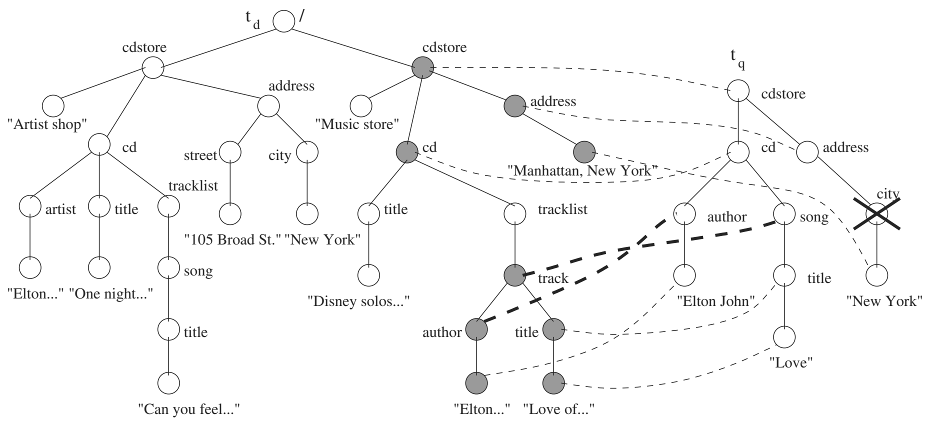


Fig. 2. Partial coverage and unbalance of query tree in `Doc1`

songs with "love" in the title. Consider Fig. 2, where a possible result is shaded in the `Doc1` tree. Note that t_q is only *partially* covered in this solution, since the `city` node does not have any correspondence in t_d. Further, the emphasized dashed lines show a *structure unbalance.*

As to point **2)**, in order to specify the ranking of results, embeddings are to be assigned a *score*, according to a *relevance ranking function* ρ.
Because of the flexibility allowed in the retrieval of embeddings, different types of relaxations are to be taken into account for relevance score computation. For instance, partiality of the $\tilde{e}$ function affects the *completeness* of results. Approximations according to the *a)* point of Def. 1 influence the *semantic correctness* of the retrieved data. Relaxations like those allowed in *b)* and *c)* denote *structural discrepancies* as well as *low cohesion* of data since intermediate nodes are allowed in the document tree. This latest relaxation is evident in Fig. 3 where the shaded solution in the `Doc2` tree presents a relevant CD in the stock "157" of the store's warehouse.
This denotes fragmentation of data retrieved, that contributes to lower the final result score. Our ranking function takes into account all these features, as detailed below.

Definition 2 (Relevance Ranking Function). We denote with ρ a *ranking function* that, given a triple $(t_q, t_d, \tilde{e}[t_q, t_d])$, returns a score $\sigma \in S = [0, 1]$ such that, with respect to the approximate embedding $\tilde{e}$ of the query expressed by t_q in the document expressed by t_d, σ is obtained as a combination of the following components:

- γ_1, that indicates how much $\tilde{e}$ is *semantically complete* with respect to the given query
- γ_2, which denotes *semantic correctness* of $\tilde{e}$, in that it states how well the embedding satisfies semantic requirements

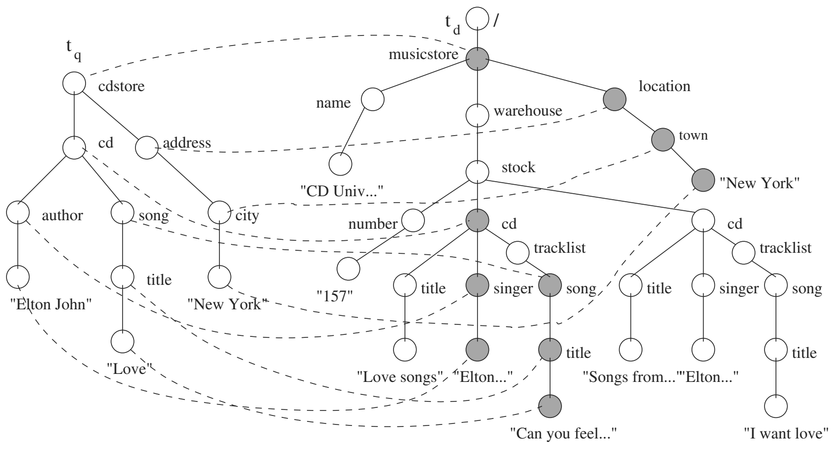


Fig. 3. Partial coverage of query tree and low cohesion in `Doc2`

- γ_3, that represents the *structural completeness* of $\tilde{e}$ with respect to the given query; it denotes the structural *coverage* of the query
- γ_4, that expresses *structural correctness* of $\tilde{e}$, in that it is a measure of how well constraints on structure are respected in the embedding
- γ_5, which specifies *cohesion* of $\tilde{e}$, by providing the grade of fragmentation of the retrieved embedding

Thus, $\sigma = \phi(\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5)$, with ϕ a *combine* function, states the *overall similarity score* of the embedding $\tilde{e}$. Formally:

$$\rho : \mathcal{T}_Q \times \mathcal{T}_D \times \mathcal{E} \to S$$

As to the components of a score σ, in Section 5 we will detail each γ_i, when discussing relevance computation. Now we are ready to introduce a *scored approximate tree embedding.*

Definition 3. (Scored Approximate Tree Embedding) A *scored approximate tree embedding* $\tilde{e}_s$ is an approximate tree embedding extended with a *score* in $\mathcal{S}$. Formally: $\tilde{e}_s : \mathcal{S} \times \mathcal{E}$.

With regard to point **3)**, consider Fig. 4. The CD store already retrieved through the embedding of Fig. 3 presents a second relevant CD. Thus, we expect this music store to be a high relevant answer to the user query, since requirements on CD occur twice. On the other hand, `Doc1` contains two relevant stores, as shown in figures 2 and 5. This also increases the relevance of this document.

In order to capture these situations, our tree embedding method considers the presence of multiple occurrences of query requirements, by providing a set-oriented approach to results. This information is then exploited to strengthen the score of *dense* results.

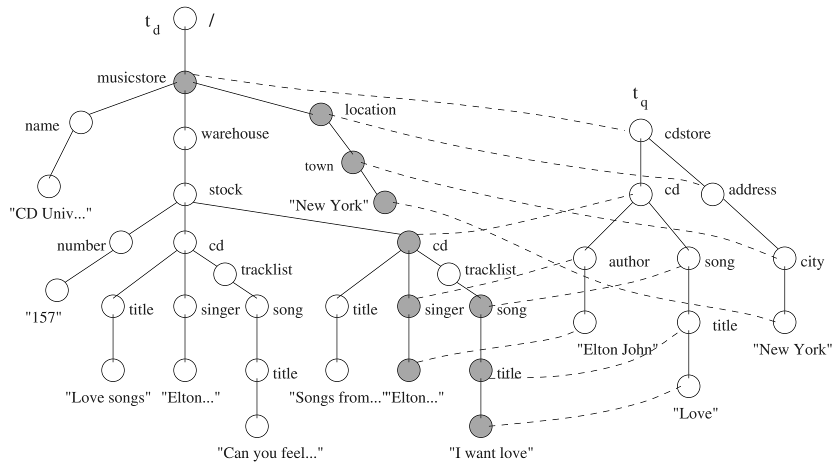


Fig. 4. Further occurrence of query tree in `Doc2`

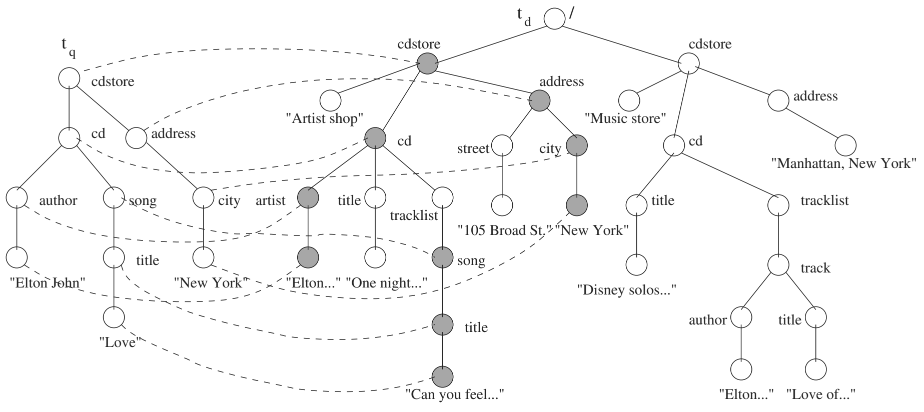


Fig. 5. Full coverage of query tree in `Doc1`

4 Approximate Embedding of Query Trees

The similarity function we propose for retrieval and scoring of embeddings of a query tree in a document tree, is given by the *SATES* (*Scored Approximate Tree Embedding Set*) *function*.

Definition 4 (*SATES* Function). We define the *Scored Approximate Tree Embedding Set Function* as:

$$SATES : \mathcal{T}_Q \times \mathcal{T}_D \rightarrow 2^{\mathcal{S} \times \mathcal{E}}$$

$\forall t_q \in \mathcal{T}_Q, \forall t_d \in \mathcal{T}_D$, $SATES(t_q, t_d)$ returns a set of scored approximate tree embeddings for t_q in t_d. The $SATES$ function returns a result set, thus capturing the possibility of having more than one embedding between a query tree and a document tree.

Intuitively, the $SATES$ function states "how well" a data tree t_d fits a query tree t_q, also taking care of multiple fittings. To determine the scored embeddings, the function is defined in a recursive fashion. In order to show the embedding process, we follow an example-driven approach, and we provide the formal definition of the $SATES$ function in appendix.

We recall the example of Fig. 2, and we try to find an embedding for t_q in t_d. For simplicity, we start considering only the second branch of the t_d tree, i.e. the one relating to `"Music store"`. We refer to it as the *current* t_d, and we denote it with t_d^*. Then, we will show later how the final embedding of t_q in t_d will include also the embedding for the `"Artist shop"` store, thus resulting in a *set* of embeddings. Starting top-down, the method tries to find a match between roots' labels. If a match is possible, then the $SATES$ function recursively applies to children of t_q and t_d^*. This is made according to a bipartite graph matching[1] between the children of t_q and the children of t_d^*, and we denote it with $\mathcal{M}_{t_q}^{t_d}$. In our example, since the labels of t_q's root and t_d^*'s root match, we consider the most promising matching $\mathcal{M}_{t_q}^{t_d}$ between the pairs ($\texttt{address}_q$,$\texttt{address}_d$), and ($\texttt{cd}_q$,$\texttt{cd}_d$), where indices q and d denote the belonging to query and data trees, respectively. In fact, other possible matchings $\mathcal{M}_{t_q}^{t_d}$, involving for instance the `"Music store"` node, do not produce relevant results. However, alternative matchings help in finding structural discrepancies, as we will show later in this Section. Then, the embedding method proceeds recursively on the abovementioned pairs.

First, let us consider the pair ($\texttt{address}_q$,$\texttt{address}_d$): We have that $t_q^* = \mathit{supp}(\texttt{address}_q)$ and $t_d^* = \mathit{supp}(\texttt{address}_d)$, with t_q^* the *current* t_q. Since roots' labels match, a $\mathcal{M}_{t_q}^{t_d}$ is established between nodes $\texttt{city}_q$ and $\texttt{"Manhattan,..."}_d$. Then, recursion is applied again. However, in this case *no similarity is found* for roots' labels (at present, $t_q^* = \mathit{supp}(\texttt{city}_q)$, and $t_d^* = \mathit{supp}(\texttt{"Manhattan,..."}_d)$). Thus, the method considers two possible strategies:

1. **(optimistic)**: it tries to find a *complete* embedding for t_q^* at a deeper level in t_d^*. This means that the search context is changed to a more specific context;
2. **(pessimistic)**: it intentionally *gives up* looking for a match of t_q^*'s root (thus accepting a *partial match*), and tries to satisfy the remaining query conditions ($t_q^* = \mathit{supp}(\texttt{"New York"}_q)$), by proceeding either:
   a) in the current data context, i.e. t_d^* stays unchanged, or
   b) changing the context, i.e. looking for a match at a deeper level in t_d^*

In the current example, it is easy to notice that the only feasible strategy is 2a. In fact, since t_d^* is indeed a *leaf*, it is not possible to change any search context,

[1] Consider a graph $G = (V, E)$. G is *bipartite* if there is a partition $V = A \cup B$ of the nodes of G such that every edge of G has one endpoint in A and one endpoint in B. A *matching* is a subset of the edges no two of which share the same endpoint. In our case $A = \mathit{children}(\mathit{root}(t_q))$ and $B = \mathit{children}(\mathit{root}(t_d^*))$.

thus making alternatives 1 and 2b not applicable. Then recursion concludes for this branch (i.e. for the pair ($\texttt{address}_q$,$\texttt{address}_d$)), since the current trees are actually two leaves, and a match is found for them.

Now, let us get back to the pair ($\texttt{cd}_q$,$\texttt{cd}_d$): We have that their labels match, and two matchings $\mathcal{M}_{t_q}^{t_d}$ are possible among their children:

1. Let consider first $\mathcal{M}_{t_q}^{t_d} = \{$($\texttt{author}_q$,$\texttt{title}_d$),($\texttt{song}_q$,$\texttt{tracklist}_d$)$\}$. It is easy to show that the former pair does not produce any result; on the other hand, in the latter one we follow strategy 1, and we recursively compute $SATES$ on the pair ($\texttt{song}_q$,$\texttt{track}_d$). This time, labels are similar and the embedding process goes on, matching ($\texttt{title}_q$,$\texttt{title}_d$), since the pair ($\texttt{title}_q$,$\texttt{author}_d$) does not return any result. Finally, the pair ("Love"$_q$,"Love of ..."$_d$) is added to the embedding.
2. As to the second possible matching $\mathcal{M}_{t_q}^{t_d} = \{$($\texttt{song}_q$,$\texttt{title}_d$), ($\texttt{author}_q$,$\texttt{tracklist}_d$)$\}$, once again we neglect the former pair. In fact, although an embedding is retrieved, i.e. $\{$($\texttt{title}_q$,$\texttt{title}_d$)$\}$, the corresponding leaves do not match, thus making the embedding not significant.[2] With regard to the second pair, the double application of strategy 1 allows for discovering the embedding of ($\texttt{author}_q$,$\texttt{author}_d$), including the pair ("Elton John"$_q$,"Elton John"$_d$). On the other hand, similarly to the previous case, the pair ($\texttt{author}_q$,$\texttt{title}_d$) leads to empty results.

Thus, in order to capture structural unbalances, the $SATES$ function takes into account all possible matchings $\mathcal{M}_{t_q}^{t_d}$. Finally, for the sake of completeness, recall that the starting embedding was $SATES(t_q, t_d)$. The set-oriented approach of our method allows for returning two possible solutions, through the double application of strategy 1: The former time to try the embedding of the pair ($\texttt{cdstore}_q$,$\texttt{cdstore}'_d$), and the latter one for the pair ($\texttt{cdstore}_q$,$\texttt{cdstore}''_d$), where the number of apices denotes the first and second CD store in Doc1, respectively. A further situation where the set-oriented approach contributes in the retrieval of multiple occurrences of results is shown in figures 3 and 4, where the final embedding set is obtained by the union of the embeddings originated by the pairs ($\texttt{cd}_q$,$\texttt{cd}'_d$) and ($\texttt{cd}_q$,$\texttt{cd}''_d$), respectively.

5 Relevance Computation

Most approaches [4,7,10,17,20] score results with ranking values that, individually, do not provide information on the *query rate* satisfied by an answer. Assume, for instance, to compare results coming from two different queries q_1 and q_2. According to most scoring methods [4,17], it is possible that one document that satisfies 1 condition (out of 2) of q_1 is assigned the same score of a document

[2] This policy privileges embeddings with query leaves matching. This is to limit the huge amount of results that satisfy structural constraints, but do not match content conditions. From a strict structural point of view, the above pair should belong to the embedding.

that satisfies 9 conditions (out of 10) of the more complex query q_2. Although results are incomparable, one would expect documents (exactly) satisfying a high percentage of conditions to score higher than documents (exactly) satisfying a lower rate. Thus, besides information on *correctness* of results, a measure of *completeness* is desirable. As to XML documents, this information is somehow made more complex by the presence of structure inside documents. This implies that some knowledge on completeness is supposed to provide also information on the matching rate of query structure. Then, apart from queries where the user explicitly specifies not to take care of the depth where information may be found in, also *cohesion* of data retrieved is an important element to be considered when ranking results.

Here, we provide a scoring method that captures the above-mentioned features. We start modelling a set of properties for each embedding $\tilde{e}$. Property values are normalized in the interval $[0, 1]$, where values close to 1 denote high satisfaction. Properties are:

Semantic Completeness. It is a measure of how much the embedding is semantically complete with respect to the given query. It is computed it as the ratio between the number of query nodes in the embedding, $n_q^{\tilde{e}}$, and the total number of query nodes, n_q:

$$\gamma_1 = \frac{n_q^{\tilde{e}}}{n_q}$$

Semantic Correctness. It states how well the embedding satisfies semantic requirements. It represents the overall semantic similarity captured by the nodes in the embedding. This is computed as a combination of label similarities of matching nodes, possibly lowered by type mismatchings (attribute vs. element nodes):

$$\gamma_2 = \bigwedge_{q_i \in dom(\mathcal{E})} sim(label(q_i), label(\tilde{e}(q_i)))$$

where $\wedge$ is a scoring function [9] the computes the conjunction of label similarities. For instance, $\wedge$ could be a fuzzy *t-norm* [13], such as the *min* function or the product operator.

Structural Completeness. It represents the *structural coverage* of the query tree. It is computed as the ratio between: 1) the number of node pairs in the image of the embedding, $hp_q^{\tilde{e}}$, that satisfy the same hierarchical[3] relationship of the query node pairs which are related to, and 2) the total number of hierarchy-related pairs in the query tree (hp_q):

$$\gamma_3 = \frac{hp_q^{\tilde{e}}}{hp_q}$$

Structural Correctness. It is a measure of how many nodes respect structural constraints. It is computed as the complement of the ratio between the

[3] Either parent-child or ancestor-descendant relationship.

number of structural penalties p (i.e. unbalances) and the total number of hierarchy-related pairs in the data tree that also appear in the embedding:

$$\gamma_4 = 1 - \frac{p}{hp_d^{\tilde{e}}}$$

Cohesion of results. It represents the grade of fragmentation of the resulting embedding. It is computed as the complement of the ratio between the number of intermediate data nodes $in_d^{\tilde{e}}$ among the nodes in the embedding, and the total number of data nodes in the embedding, also including the intermediate ones, $n_d^{\tilde{e}}$:

$$\gamma_5 = 1 - \frac{in_d^{\tilde{e}}}{n_d^{\tilde{e}}}$$

These properties can be naturally partitioned in two sets: properties related to semantics, and properties concerning structure. The combination of the first two scores provides the overall information on *semantic satisfaction*. As to the remaining properties, they all provide different perspectives for the evaluation of structure similarity. A combination of them indicates a global measure, that summarizes the *structural satisfaction* of the retrieved data. However, it is beyond the scope of this paper to evaluate which is the best function to be used for combining these scores in the computation of the overall score σ of an embedding, as defined in Def. 2. Clearly, additional flexibility can be reached by assigning *weigths* to each γ_i to denote the different importance of each property.

5.1 Comparison with Related Approaches

Table 1 shows a comparison of our ranking method with other similar approaches, according to different relaxations on structure.

Table 1. Relaxations supported on structure

Approach	partial match	unbalance	intermed. data nodes	multiple occurr.
XXL [20]	no	no	yes	no
XIRQL [10]	no	no	yes	no
Damiani et al. [7]	no	no	yes	no
Amer-Yahia et al. [4]	no	yes[4]	yes	no
ApproXQL [17]	yes	no	yes	no
SATES	yes	yes	yes	yes

All the discussed methods allow for intermediate data nodes in the results. However, all but ApproXQL do not consider the number of exceeding data nodes in the computation of scores. In fact, all methods except ApproXQL would return

[4] *Subtree Promotion* in [4] captures the unbalances described in Def. 1, point *b)*. The symmetric case, point *c)*, is not discussed.

the same score for the approximate tree embeddings of figures 4 and 5, even though the former embedding presents a higher fragmentation.[5]

However, consider the query embedding of Fig. 6, where the current query is simplified with respect to the sample we used throughout the paper, since it does not require any constraint on CD store location.

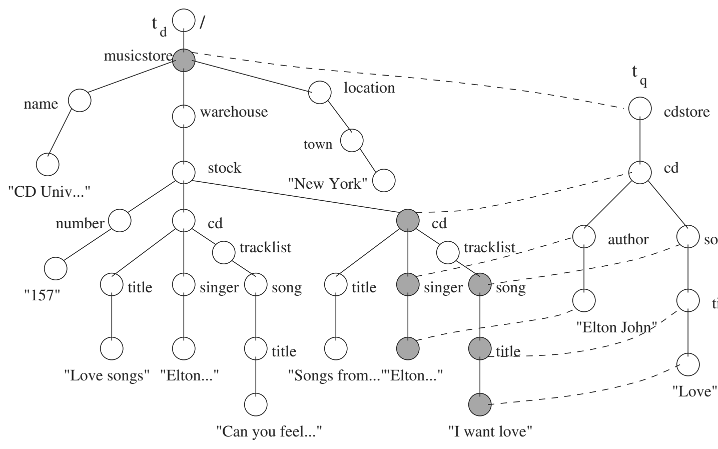


Fig. 6. Embedding of the simplified sample query

Table 2 shows how ApproXQL and *SATES* evaluate the embeddings of figures 4 and 6. For ApproXQL, the same score would be assigned in both cases, since the

Table 2. Different scoring in presence of different cohesion of data

Approach	Score for embedding of Fig. 4	Score for embedding of Fig. 6
ApproXQL	high	high
SATES	high	medium

number of relaxations to meet data organization can be quantified in the same number of intermediate nodes for both queries (3 nodes, indeed). Nevertheless, the set of query requirements is different, and the number of exceeding nodes is proportionally more heavy for the embedding of Fig. 6. *SATES* takes into account this feature, and assigns two different scores to results, scoring higher the embedding of Fig. 4.

[5] For simplicity, we assume the matchings on labels to score the same for both embeddings, so that only evaluation on structure affects the ranking.

6 Implementation

We implemented a prototype and experienced our similarity measure on a real collection of XML documents. The dataset used is provided by the Astronomical Data Center [1]. The prototype has been developed in Java2 v.1.3.1 and uses the API for XML parsing, and Xerces 1.4.4 to parse documents. As to semantic similarity, the system refers to the WordNet semantic network, [2] exploiting relationships like synonymy, hyperonymy, and holonymy among terms, and accesses it through the Java WordNet Library (JWNL). As a similarity measure we used an adaptation of the Sussna's formula [5]. We plan to experience and refine our method, in order to find the better rules to be used for our *combine* and *ranking* functions, with the aim of representing results at the best of their meaning.

7 Conclusions and Future Work

Most of the existing ranking approaches are inadequate when querying heterogeneous collections of XML documents. In fact, as to similarity on documents' structure, they basically require *all* query conditions are somehow preserved in the data retrieved. Also, they do not fully exploit relaxations on structure to capture slightly different organization of data.

We presented an approach that widens the spectrum of relevant data, including solutions that also partially satisfy query requirements, and that approximate text organization inside documents, also capturing unbalances of structure and multiple occurrences of query conditions. Then, we rank results according to a set of properties of the retrieved data. These properties provide, besides correctness, a measure of completeness of query satisfaction, as well as knowledge about cohesion of results. In order to augment query flexibility we plan to allow the user to express preferences on either the semantics or the structure of a query. As to queries with conditions referring to ordered data, we intend to study constraints on our generic unordered tree embedding method. We are aware of the problem of blindly querying XML data: Documents might be organized largely differently from the user's point of view. Thus, we plan to study new complex hierarchical relaxations on data, and to make this process transparent to the user. With regard to implementation, in order to cope with the possible hugeness of approximate results returned, we plan to use threshold conditions, to keep only the most relevant results.

References

1. ADC XML Resources Home Page. `http://xml.gsfc.nasa.gov/`.
2. WordNet Home Page. `http://www.cogsci.princeton.edu/ wn/`.
3. XML Information Set. `http://www.w3.org/TR/xml-infoset`.
4. S. Amer-Yahia, S. Cho, and D. Srivastava. Tree Pattern Relaxation. In *Proc. of the 8th Int. Conf. on Extending Database Technology (EDBT 2002)*, March 2002.

5. A. Budanitsky. Lexical Semantic Relatedness and its Application in Natural Language Processing. Technical Report CSRG-390, University of Toronto, 1999.
6. S. Ceri, S. Comai, E. Damiani, P. Fraternali, S. Paraboschi, and L. Tanca. XML-GL: A Graphical Language for Querying and Restructuring XML Documents. *Computer Networks*, 31:1171–1187, 1999.
7. E. Damiani and L. Tanca. Blind Queries to XML Data. In *In Proc. of Int. Conf. on Database and Expert Systems Applications (DEXA)*, pages 345–356, 2000.
8. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: A Query Language for XML. In *Proc. Int. World Wide Web Conference)*, Canada, 1999.
9. R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of the 15th ACM Symposium on Principles of Database Systems (PODS'96)*, pages 216–226, Montreal, Canada, June 1996.
10. N. Fuhr and K. Großjohann. XIRQL: A Query Language for Information Retrieval in XML Documents. In *Proc. ACM SIGIR Conference*, 2001.
11. R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. In *In Proc. of 2nd Int. Workshop on the Web and Databases (WebDB'99)*, Philadelphia, PA, June 1999.
12. P. Kilpeläinen. *Tree Matching Problems with Application to Structured Text Databases*. PhD thesis, Dept. of Computer Science, Univ. of Helsinki, SF, 1992.
13. G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall PTR, 1995.
14. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *Proc. of the 27th VLDB Conf.*, pages 49–58, Rome, Italy, 2001.
15. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In *Proc. of the 18th Int. Conf. on Data Engineering (ICDE 2002)*, San Jose, CA, March 2002.
16. J. Robie, L. Lapp, and D. Schach. XML Query Language (XQL). In *Proc. of the Query Language Workshop (QL'98)*, Cambridge, Mass., 1998.
17. T. Schlieder. Similarity Search in XML Data Using Cost-Based Query Transformations. In *Proc. of 4th Int. Work. on the Web and Databases (WebDB01)*, 2001.
18. T. Schlieder and H. Meuss. Result Ranking for Structured Queries against XML Documents. In *DELOS Workshop: Information Seeking, Searching and Querying in Digital Libraries*, 2000.
19. T. Schlieder and F. Naumann. Approximate Tree Embedding for Querying XML Data. In *In Proc. ACM SIGIR Workshop on XML and Information Retrieval*, Athens, Greece, July 2000.
20. A. Theobald and G. Weikum. Adding Relevance to XML. In *Proc. 3rd Int. Workshop on the Web and Databases (WebDB 2000)*, pages 35–40, May 2000.
21. J. Wolff, H. Florke, and A. Cremers. Searching and Browsing Collections of Structural Information. In *Proc. of the IEEE Advances in Digital Libraries*, pages 141–150, USA, May 2000.

A The *SATES* Function

For simplicity, let $sim(t_q, t_d)$ be $sim(label(root(t_q)), label(root(t_d)))$, and let $s(t_q, t_d) = \rho(t_q, t_d, \{root(t_q), root(t_d)\})$. We use $\mathcal{M}$ in place of $\mathcal{M}_{t_q}^{t_d}$. The $\ominus_q$ and $\ominus_d$ functions change the scores of a set of approximate tree embeddings, according to a lowering factor. New scores capture the unsuccessful match of a query node and a data node, respectively. The $\otimes$ function generates a new score from a set of n given scores in $\mathcal{S}$.

```
$\forall t_q \in \mathcal{T}_Q, t_d \in \mathcal{T}_D, SATES(t_q, t_d)$ is defined as:

case $leaf(root(t_q)) \wedge leaf(root(t_d))$:
    if $sim(t_q, t_d) > 0$
      $SATES(t_q, t_d) = \{[(s(t_q, t_d), \{(root(t_q), root(t_d))\}]\}$
    else $SATES(t_q, t_d) = \emptyset$

case $leaf(root(t_q)) \wedge \neg leaf(root(t_d))$:
    if $sim(t_q, t_d) > 0$
      $SATES(t_q, t_d) = \{[(s(t_q, t_d), \{(root(t_q), root(t_d)\})]\}$
    else $SATES(t_q, t_d) = \bigcup_{c \in children(t_d)} \ominus_d \ (SATES(t_q, supp(c)))$

case $\neg leaf(root(t_q)) \wedge leaf(root(t_d))$:
    if $sim(t_q, t_d) > 0$
      $SATES(t_q, t_d) = \{[(s(t_q, t_d), \{(root(t_q), root(t_d)\})]\}$
    else $SATES(t_q, t_d) = \bigcup_{c \in children(t_q)} \ominus_q \ (SATES(supp(c), t_d))$

case $\neg leaf(root(t_q)) \wedge \neg leaf(root(t_d))$:
    if $sim(t_q, t_d) > 0$ $SATES(t_q, t_d) =$
    $\bigcup_{\mathcal{M}_{t_q}^{t_d}} \bigcup_{\substack{(t_i^k, t_j^k) \in \mathcal{M} \\ (s_{l_k}^k, m_{l_k}^k) \in SATES(t_i^k, t_j^k) \\ l_k \in [1..|SATES(t_i^k, t_j^k)|]}} [\otimes(s(t_q, t_d), s_{l_1}^1, \ldots, s_{l_{|\mathcal{M}|}}^{|\mathcal{M}|}), \{(root(t_q), root(t_d))\} \cup m_{l_1}^1 \cup \ldots \cup m_{l_{|\mathcal{M}|}}^{|\mathcal{M}|}]$
    else $SATES(t_q, t_d) = \bigcup(\bigcup_1, \bigcup_2, \bigcup_3)$
              where $\bigcup_1 = \bigcup_{\mathcal{M}_{t_q}^{t_d}} \ominus_q \circ \ominus_d \bigcup_{\substack{(t_i^k, t_j^k) \in \mathcal{M} \\ (s_{l_k}^k, m_{l_k}^k) \in SATES(t_i^k, t_j^k) \\ l_k \in [1..|SATES(t_i^k, t_j^k)|]}} [\otimes(s_{l_1}^1, \ldots, s_{l_{|\mathcal{M}|}}^{|\mathcal{M}|}), m_{l_1}^1 \cup \ldots \cup m_{l_{|\mathcal{M}|}}^{|\mathcal{M}|}]$
                    $\bigcup_2 = \bigcup_{c \in children(t_d)} \ominus_d \ (SATES(t_q, c))$
                    $\bigcup_3 = \bigcup_{c \in children(t_q)} \ominus_q \ (SATES(c, t_d))$
```

An Efficient Incremental Lower Bound Approach for Solving Approximate Nearest-Neighbor Problem of Complex Vague Queries

Tran Khanh Dang, Josef Küng, and Roland Wagner

FAW Institute
Johannes Kepler University of Linz, Austria
{khanh, jkueng, rwagner}@faw.uni-linz.ac.at

Abstract. In this paper, we define a complex vague query as a multi-feature nearest neighbor query. To answer such queries, the system must search on some feature spaces individually and then combine the results in order to find the final answers. The feature spaces are usually multidimensional and may consist of the sheer volume of data. Therefore searching costs are prohibitively expensive for complex vague queries. For only such a single-feature space, to alleviate the costs, problem of answering nearest neighbor and approximate nearest neighbor queries has been extensively studied and quite well addressed in the literature. This paper, however, introduces an approach for finding (1+ε)-approximate nearest neighbors of complex vague queries, which must deal with the problem on multiple feature spaces. This approach is based on a novel, efficient and general algorithm called ISA-Incremental hyper-Sphere Approach [12, 13], which has recently been introduced for solving nearest neighbor problem in the VQS-Vague Query System [22]. To the best of our knowledge, the work presented in this paper is one of the vanguard solutions for generally dealing with problem of approximate multi-feature nearest neighbor queries. The experimental results will prove the efficiency of the proposed approach.

1 Introduction

Similarity search has emerged and become a fundamental paradigm for a variety of application areas as multimedia [30, 27], data mining [4], CAD database systems [5], time-series databases [15], information retrieval (IR) systems [2], geographical information system and tourist information system [29], etc. In these contexts an important problem is how to find the data object(s) that is most similar to a given query object. The standard approach is to perform search on feature vectors repositories of the data objects and use specific computational functions, e.g. Euclidean metric, to measure the distance between feature vectors. Similarity search problem therefore becomes a nearest neighbor (NN) query over the feature vector spaces. Notably, there are some proposals to extend and facilitate the conventional relational database management systems (DBMSs) with similarity search capabilities [19, 25, 22, 8]. Specially, in [22] Kueng et al. proposed an approach, called Vague Query System (VQS), to solving the similarity search problem in the conventional DBMS context but it also has aspects similar to that of the approaches to modern data repositories mentioned above. More concretely, the VQS is an extension to the conventional DBMSs and can operate on top of them to return tuples semantically close to user's query. The VQS key feature is concept of Numeric-Coordinate-

Representation-Tables (NCR-Tables) that store semantic information of attributes. In fact, attributes of arbitrary types in a query relation/view are mapped to the Euclidean spaces and kept by NCR-Tables. When the DBMS fails to find an exact match for a given query object Q, the VQS will search on some NCR-Tables corresponding to the query conditions of Q and return the best match for Q. Intuitively, NCR-Tables in the VQS are equivalent to the feature vector spaces above.

Solving NN queries has been researched for a long time and the de factor standard solution is to employ multidimensional access methods (MAMs) [16, 3] to index feature values, which are often multidimensional vectors. Searching is later done with support of MAMs to speed-up the performance and to decrease the costs, i.e. IO-cost and CPU-cost. Nevertheless, when the number of dimensions of the feature spaces increases, MAMs typically suffer from the so-called dimensionality curse. This phenomenon has been observed and shown that in high-dimensional feature spaces, e.g. greater than 16, the performance of MAMs significantly decreases and a linear scan over the whole data set would perform better [32, 6].

In practice, the mapping of attributes of objects to coordinates of vectors is heuristic in many application domains [21]. Therefore an approximate similarity search is in many cases as good as an exact search or a "true" similarity search. Given a data set S and a query object Q, a point P is called a (1+ε)-approximate nearest neighbor of Q with ε>0 if for all P'∈ S:

$$\mathrm{dist}(Q, P) \leq (1+\varepsilon)\mathrm{dist}(Q, P') \,. \tag{1}$$

In inequality as shown in equation 1, dist(X, Y) represents the distance between objects X and Y. There are many previous researches that have dealt with approximate NN queries. However, these proposed approaches have solved problem of finding approximate NNs of a given query either over only a single feature space [1, 20, 21, 28, 9] or for different matters [33, 34] from the VQS's. Although there are also many researches into multi-feature NN queries answering as [14, 27, 26, 8, 17, 7], we cannot find any solution to adapt it for solving approximate multi-feature NN queries *in the VQS*. Recently, we have introduced an approach named Incremental hyper-Sphere Approach (ISA) [12, 13] that has been shown to be efficient and general for supporting similarity search capabilities in the conventional DBMSs. The ISA is an improvement and a generalization of an approach proposed for the VQS in [24][1]. Specially, the ISA can be applied to searching on multiple feature spaces that do not satisfy three conditions so that the Fagin's algorithm [14] and all its improvements [26, 17] become applicable. These conditions are as follows [31]: (1) there is at least a key for each feature space to be searched, (2) there is a mapping between the keys, and (3) we must ensure that the mapping is one-to-one. Intuitively, the condition (1) is always satisfied in the VQS, however, the condition (3) does not hold. In the VQS, each *Fuzzy Field* is also the key for the corresponding NCR-Table but there is no the mapping one-to-one between *Fuzzy Fields* of the NCR-Tables. In this paper we will present an approach based on the ISA to efficiently and generally solving approximate multi-feature NN queries, which are also called approximate complex vague queries (CVQs).

[1] Here we must note that although the ISA looks similar to the J* algorithm presented by Natsev et al. [34], they are intrinsically different. Also, the ISA was inspired by the Incremental hyper-Cube Approach introduced by Kueng et al. [24], which had been introduced *earlier* and had the same overall goals as the J* algorithm.

The rest of the paper is organized as follows. Section 2 summarizes the ISA, which is the base of approach presented in this paper. Section 3 is dedicated to introducing ε-ISA: an incremental lower bound approach for efficiently finding approximate NN of CVQs. Section 4 presents experimental results that prove the efficiency of the ε-ISA. Section 5 gives conclusions and future work.

2 ISA: An Efficient Approach for Solving Complex Vague Queries

Query processing in the conventional DBMSs is not flexible enough to support similarity search capabilities directly. That means when the available data in a relational database does not match a user's query precisely, the system will only return an empty result set to the user. This limits their applicability to domains where only crisp answers are meaningful. In many other application domains, however, the users also expect not only the crisp results returned but also some other results close to the query in a sense. Systems can solve this problem are called Flexible Query Answering Systems (FQASs) [13]. Among FQASs proposed (cf. section 1) the VQS [22] is very notable because it can be easily built, extensible and workable on top of the existing DBMSs. Specially the VQS employs NCR-Tables to store semantic information of attributes of a query relation/view and the similarity search is carried out with their support. This approach resembles that of modern IR systems [2]. In [24] Kueng et al. presented an Incremental hyper-Cube Approach (ICA) for finding the best match for multi-feature NN (M-FNN) queries in the VQS. Unfortunately, the ICA is not general and has weaknesses lead to degenerate the search performance of the system. Recently, in [12, 13] we introduced an Incremental hyper-Sphere Approach (ISA) to improve and generalize the ICA for solving M-FNN queries. One of the ISA key features is to use hyper-sphere range queries instead of hyper-cube ones as the ICA for reducing I/O-cost and CPU-cost during the search process. Figure 1 illustrates the scenario. Assume the example feature space in figure 1 is involving a query condition q_i of the running CVQ and the first *appropriate tuple* of the query relation/view is found after carrying out hyper-cube range query h_{i0} (an appropriate tuple in context of the ICA is one that belongs to the query relation/view and contains the NCR-Values returned with respect to each involved query condition; see [24] for details of the ICA). To verify if this appropriate tuple is the best match for the query, the ICA must extend the searching radius and create a new hyper-cube query h_{i1}. Intuitively, objects located in the grey area $s=h_{i1}-c_{i1}$ are unnecessary to be verified but the ICA does it while executing query h_{i1}. For the ISA, only objects in hyper-sphere c_{i1} are accessed.

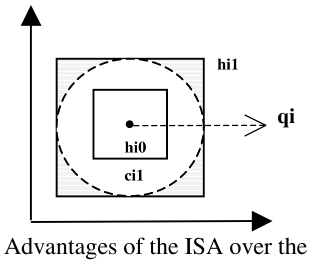


Fig. 1. Advantages of the ISA over the ICA

Input: A query relation/view S; a complex vague query Q with n query conditions q_i (i=1, 2… n); and assume that each feature space involving the query Q is managed by a multidimensional index structure F_i.

Output: Best match record/tuple T_{min} for Q, $T_{min} \in S$. Ties are arbitrarily broken.

Step 1. Search on each F_i for the corresponding q_i using the adapted incremental algorithm for hyper-sphere range queries.

Step 2. Combine the searching results from all q_i to find at least an appropriate record in S, which contains the NCR-Values returned with respect to each query condition. If there is no appropriate record found, go back to step 1.

Step 3. Compute total distances/scores for the found records using formula 2 below and find a record T_{min} with the minimum total distance TD_{cur}. Ties are broken arbitrarily.

$$TD = \left(\sum_i (di/Di) * wi \right) / wsum \ . \qquad (\mathbf{2})$$

where:

TD	total distance/score of a record
D_i	diameter of feature space i
w_i	weight of query condition q_i
w_{sum}	sum of all weights w_i
d_i	distance from q_i to the returned NCR-Value in the corresponding F_i

Step 4. Compute the maximum searching radius r_{inew} for each q_i with respect to TD_{cur} using formula 3 below and continue doing the search as steps 1, 2 and 3 until one of two following conditions holds: (a) the current searching radius of each q_i is greater than or equal to its maximum searching radius; (b) found a new appropriate record T_{new} with the total distance $TD_{new}<TD_{cur}$.

$$r_{inew} = D_i * TD_{cur} * w_{sum} / w_i \ . \qquad (\mathbf{3})$$

where:

r_{inew}	new (maximum) searching radius of q_i
D_i, w_i, w_{sum}	defined as in formula 2

Step 5. If condition (a) holds, return T_{min} as the best match for Q. Otherwise, i.e. condition (b) holds, replace T_{min} with T_{new}, i.e. TD_{cur} is also replaced with a smaller value TD_{new}, and go back to step 4.

Fig. 2. Incremental hyper-Sphere Approach (ISA) for answering Complex Vague Queries

Moreover, for each NCR-Table (i.e. feature space) involving the query, the ISA employs an incremental algorithm for range queries, which is modified from one of the state-of-the-art algorithms for solving k-NN queries presented in [18]. This modified algorithm is also proven to be optimal in terms of disk access number as the original one [13]. Figure 2 briefly summarizes high-level abstract steps of the ISA[2].

After doing step 1 there are several returned NCR-Values for each query condition q_i. Here the incremental algorithm adapted for range queries is used to avoid repetitive disk accesses when processing the range queries. This is illustrated in figure 3: During the execution of range queries q_{i1}, q_{i2} the algorithm does not have to access again disk pages accessed by range queries q_{i0}, q_{i1}, respectively. This is one of the advantages of the ISA over the ICA: In figure 1, for example, to process range query h_{i1} the ICA must access again all disk pages accessed while doing range query h_{i0}.

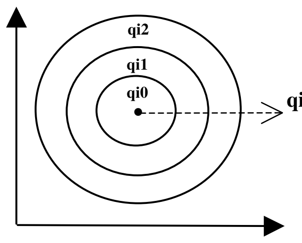


Fig. 3. Incremental algorithm adapted for range queries

Step 2 of the algorithm tries to find an appropriate record/tuple, which belongs to the query relation/view and contains the returned NCR-Values in step 1 for each query condition. Step 3 is intuitive: Among appropriate records found in step 2, the ISA chooses one, i.e. T_{min}, having minimum total distance TD_{cur}, which is calculated by formula 2. Depending on TD_{cur} the ISA computes the maximum searching radius for each query condition q_i using formula 3. The search is continued and to reduce the searched space for each q_i, these maximum searching radii are updated dynamically whenever a new record T_{new} is found and its total distance $TD_{new}<TD_{cur}$, i.e. condition (b) in step 4 holds. In this case, T_{new} replaces T_{min} and the search is then kept on until condition (a) in step 4 satisfies. At this time the current T_{min} is returned in step 5. Note that when condition (a) in step 4 does not hold for a certain q_i then the search is still continued for that q_i. Obviously, after step 5 there is no any record in the entire query relation/view S having a smaller total distance to Q than that of the returned record T_{min}. Therefore, the ISA correctly returns the best match for the query Q and totally overcomes all shortcomings of the ICA.

Although experimental results in [12, 13] have shown that the ISA is significantly outperforms the ICA introduced in [24] by orders of magnitude in both terms of I/O-cost and CPU-cost, its performance is still decreased when dimension number of the feature spaces or query condition number increases. For high dimensional feature spaces, the problem is caused by the dimensionality curse, in which probability of overlaps between a query and data regions is very high, and thus query processing requires more I/O-cost and CPU-cost. For the second reason, i.e. the query condition number, figure 4 shows experimental results of 100 randomly selected three-condition queries in only two-dimensional spaces for both ICA and ISA (See section 4 for more information on test conditions). We can observe that the affected object number in the

[2] See [12, 13] for the detailed discussion as well as related information

ISA, which is directly proportional to CPU-cost [12], is not much less than that of the ICA, and thus the ISA's performance is decreased. This problem is more serious when the query condition number is larger. To face these problems we propose a variant of the ISA called ε-ISA to approximately answer CVQs (or M-FNN queries). Section 3 will detail the ε-ISA.

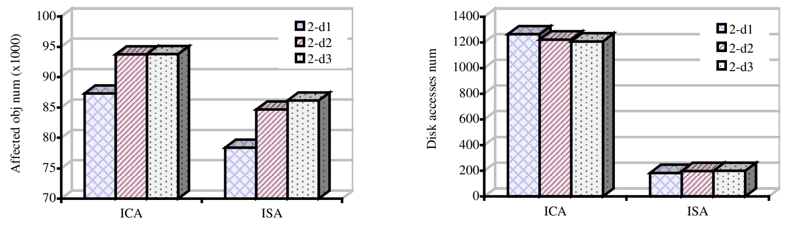


Fig. 4. Affected object and disk access number

3 ε-ISA: Towards Efficiently Addressing Approximate Complex Vague Queries

Inequality as shown in equation 1 has been applied to solving approximate single-feature nearest neighbor (S-FNN) queries [1]. Here we define a similar inequality that will be used for dealing with M-FNN queries as follows:

Definition 1. Given a M-FNN query Q of n query conditions, a query relation (or a query view) S and NCR-Tables corresponding to the mapped attributes of S. Assume the total distance (TD) from each record of S to Q is calculated as formula 2. A record $T_{app} \in S$ is defined as a (1+ε)-approximate nearest neighbor of Q with ε>0 if for all records $T \in S$ the total distances satisfy the following inequality:

$$TD(Q, T_{app}) \le (1+\varepsilon) TD(Q, T) . \quad (4)$$

Inequality as shown in equation 4 looks like inequality as shown in equation 1, however the differences are inside them because a CVQ may consist of several S-FNN queries. Each query condition of a CVQ is separately addressed and the distance metric can be different among them. Nevertheless, returned distances of objects corresponding to each query condition are then normalized into the interval [0, 1], i.e. the factor (d_i/D_i) in formula 2. The total distance TD of each record to the query Q, which is calculated as formula 2, is the weighted sum of the normalized distances just mentioned above. In accordance with the ISA presented in figure 2, an appropriate record T_{min} having a minimum TD among the found records must be verified if it is the nearest neighbor of Q. With the approximate problem, however, if its total distance TD_{cur} is satisfied inequality as shown in equation 4 then it will be returned immediately as a (1+ε)-approximate nearest neighbor of Q without more searches. Section 3.1 will present a variant of the ISA using *lower bound total distance (LBTD)* of the nearest neighbor of Q to deal with this problem.

3.1 Finding Approximate Nearest Neighbor

Figure 5 describes high-level processing steps of the ε-ISA algorithm. They are quite similar to those of the ISA described in figure 2 except for some differences as follows. First, the input information includes a real $\varepsilon>0$ that is served as a *tolerant error* to judge whether or not a found appropriate record is a $(1+\varepsilon)$-approximate NN of the given query Q. Second, the output of the ε-ISA is an approximate NN instead of the true NN of the query. As the ISA, from step 1 to step 3 the ε-ISA tries to find an appropriate record in the query relation/view S. The record T_{app} in step 3 has the minimum TD to the query among the found appropriate records. At this point we still do not know if T_{app} is a $(1+\varepsilon)$-approximate NN of Q. However, in step 4 the ε-ISA computes a LBTD for the "true" NN of Q as follows:

$$\text{LBTD} = \min\{\text{TD}_{cur}, d_1, d_2, \ldots d_n\} \qquad \textbf{(5)}$$

In formula 5, n is the query condition number of the involved query, and each d_i (i=1, 2… n) represents distance from the query condition q_i to the *farthest object* (i.e. NCR-Value) returned so far in the corresponding feature space. Because the incremental algorithm adapted for range queries employs a priority queue [12], so d_i is also distance from q_i to the *last returned object* so far in the corresponding feature space. The main idea here is to find a value LBTD ensured that it is always less than or equal to the distance from the true NN to the query Q. Besides, this LBTD should make the algorithm as much cost-effective as possible. Ideally, LBTD would take a value that equals the distance from the true NN to the query Q. Figure 6 depicts an example for this idea with a 2-condition M-FNN query.

In figure 6, assume that the query Q of two conditions q_1 and q_2 is executing and the sequential range queries for q_1 and q_2 are q_{10}, q_{11}, q_{12} and q_{20}, q_{21}, q_{22}, individually. These range queries are carried out in step 1 of the algorithm. Assume that the solid line connecting two objects of q_{11} and q_{20} represents an appropriate record T_{app} found after step 3 of the ε-ISA. That means two attributes of the record T_{app} contain two corresponding NCR-Values returned from range queries q_{11} and q_{20}, and TD_{cur} is the minimum TD among TDs of the appropriate records that have been found. Moreover, we also suppose that the ε-ISA is now doing range queries q_{12} and q_{22}. Intuitively, if T_{app} is not the "true" NN of Q then the total distance of the "true" NN cannot be less than the searching radii of q_{12} and q_{22}. This total distance is equal to the radius of q_{12}, for example, when the appropriate record found is an exact composition of q_2 and a returned NCR-Value for the range query q_{12} (see dashed line T_1). A similar explanation is for dashed line T_2. Therefore, in figure 6, $\text{LBTD}=\min\{\text{TD}_{Tapp}, \text{TD}_{T1}, \text{TD}_{T2}\} = \min\{\text{TD}_{cur}$, radius of q_{12}, radius of $q_{22}\}$ because we do not exclude a case in which T_{app} is itself the "true" NN of Q.

Depending on the LBTD computed by step 4, step 5 of the algorithm decides whether record T_{app} is a $(1+\varepsilon)$-approximate NN of Q or not. Step 6 is quite similar to step 4 of the ISA to ensure that the ε-ISA will not access unnecessary objects and disk pages during the search process. In case if the current searching radii of every q_i are greater than or equal to their corresponding maximum searching radii then searching on every F_i is stopped and actually the returned record is the "true" NN of Q.

Input:
- A query relation/view S.
- A complex vague query Q with n query conditions q_i (i=1, 2… n).
- Assume each feature space (or NCR-Table) related to Q is managed by a multidimensional index structure F_i.
- A real $\varepsilon>0$ used as a tolerant error

Output:
- $(1+\varepsilon)$-approximate NN record/tuple T_{app} for Q, $T_{app} \in S$. Ties are arbitrarily broken.

Step 1. Search on each F_i for the corresponding q_i using the adapted incremental algorithm for hyper-sphere range queries.

Step 2. Combine the searching results from all q_i to find at least an appropriate record in S, which contains the NCR-Values returned with respect to each query condition. If there is no appropriate record found then go back to step 1.

Step 3. Compute total distances for the found records using formula 2 and find a record T_{app} with the minimum total distance TD_{cur}. Ties are broken arbitrarily.

Step 4. Let d_i be distance from query condition q_i to the last NCR-Value returned in the corresponding feature space, which is being managed by F_i. Compute LBTD=min$\{TD_{cur}, d_1, d_2 \dots d_n\}$

Step 5. If $TD_{cur} \le (1+\varepsilon)$LBTD, return T_{app} as a $(1+\varepsilon)$-approximate NN record for Q. Otherwise, go to step 6.

Step 6. Compute the maximum searching radius for each q_i with respect to TD_{cur} using formula 3 and continue doing the search as steps from 1 to 5 until the algorithm is stopped at step 5. If the current searching radius of a certain q_i is greater than or equal to its maximum searching radius then searching on F_i is stopped.

Fig. 5. The ε-ISA: Finding (1+ε)-Approximate NN of a CVQ

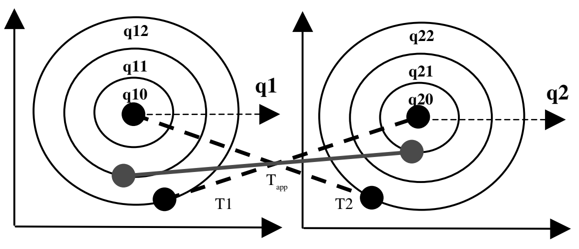


Fig. 6. Lower Bound Total Distance of a CVQ

3.2 A Generalized Algorithm for Finding Approximate k-Nearest Neighbors

The ε-ISA version described in section 3.1 just returns a (1+ε)-approximate NN of a CVQ. In practice, almost users need some approximate top-k NNs to a CVQ. We define a record/tuple is a (1+ε)-approximate k^{th} NN to a M-FNN query as follows:

Definition 2. Given a M-FNN query Q of n query conditions, a query relation/view S and NCR-Tables corresponding to the mapped attributes of S. Assume the total distance (TD) from each record of S to Q is calculated as formula 2. A record $T_{app} \in S$ is defined as a (1+ε)-approximate k^{th} NN of Q with ε>0 if the ratio of the TD between Q and T_{app} and the TD between Q and its true k^{th} NN is at most (1+ε).

In fact, in definition 2, a (1+ε)-approximate k^{th} NN of Q can be closer to Q than the true k^{th} NN [1]. Note that the incremental algorithm adapted for range queries [12], which is being used for the ε-ISA (figure 5), is originated from an algorithm for finding k NNs of a S-FNN query in which k may be unknown in advance [18]. The problem of dealing with approximate k-nearest neighbors for M-FNN queries and k is not necessary to be known beforehand is easily solved by making some modifications to the ε-ISA as follows.

1. A priority queue PQ (or a sorted buffer) is needed to keep all appropriate records to Q found in step 3 of the ε-ISA in figure 5.
2. The next approximate NN of the query Q is verified similarly to the ε-ISA version but here the record at top of PQ takes part as T_{app} in the step 3 of the algorithm.
3. When a record at the top of PQ is ensured to be the next approximate NN to the query Q, it will be added to the result set and be removed from PQ.
4. In the case PQ is empty but the user still needs more results, the search is continued normally on all the involved feature spaces. Due to the aspects of the incremental algorithm adapted for range queries, the search will continue from where it left when computing the previous approximate NN.

The correctness of this ε-ISA version is easily inferred from that of the ε-ISA in previous section. Besides, to answer approximate k-NN queries for CVQs in case the value of k is known in advance, we do not need to maintain all appropriate records to Q in a PQ as discussed above but only k closest appropriate records to Q. The other modifications are the same as mentioned. In section 4 below we show experimental results to prove the efficiency of the ε-ISA.

4 Experimental Results

To establish the practical value of the ε-ISA, we implemented it and carried out a number of experiments. Our experiments used uniformly distributed data sets of 2, 4 and 8 dimensions. Each of them had 100K objects of form <id, values list> and took role as a feature space (NCR-Table). The experiments were carried out for CVQs of 2 and 3 query conditions. For each feature space of all experiments, 100 query points were randomly selected from the corresponding data set. Furthermore, we used the

SH-tree [10] to manage the storage and retrieval of multidimensional objects in each feature space. We selected the SH-tree because it is one of the most flexible and efficient multidimensional index structures [11, 10]. All the tests were tackled on an Ultra-Sparc-II 360MHz with 512 Mbytes of main memory and some Gigabytes of hard disk capacity. All programs were implemented in C++. The page size was 8Kb for the data sets to meet with the disk block size of the operating system.

For all experiments, we simulated the database access performance to the query relation/view by comparing field id of the affected objects. For example, if a 4-dimensional object <id_4, *values list*> and an 8-dimensional object <id_8, *values list*> were retrieved during the search process and $id_4=id_8$, then they were considered belonging to a record in the query relation. This record is called an appropriate record in the paper. Intuitively, for a particular query optimization processor in the database system, the affected object number in each feature space is directly proportional to the cost of the database access (the query relation/view). Therefore we reported the disk access number of the searches on the SH-trees and the affected object number for all experiments. The affected objects here are ones located in the last hyper-sphere range query for each feature space.

Our first experiment was carried out with the data sets of 4 and 8 dimensions to find (1+ε)-approximate NN for 2-condition M-FNN queries. The ε values selected for the tests were 0.2, 0.4, 0.6, 0.8 and 1. Figure 7, 8 shows percentage of disk access savings and affected object savings, individually, for each feature space against the ε values; and figure 9 depicts the accuracy of the approximate results, compared to the true NNs, against the ε values. We observe that the accuracy is very high while all costs are significantly saved. For example, for ε=0.6 (that means the tolerant error is allowed up to 60%), the ε-ISA can save over 55% and nearly 7% disk accesses, nearly 80% and over 35% affected object number of 4- and 8-dimensional feature space, individually, while it still preserves the accuracy at 90%. The experimental results are also promising for the small ε value, e.g. ε=0.2 as depicted in figure 9 the accuracy is up to 98% and the cost savings are still quite high, e.g. over 27% and over 44% for disk accesses and affected objects of the 4-dimensional feature space, respectively.

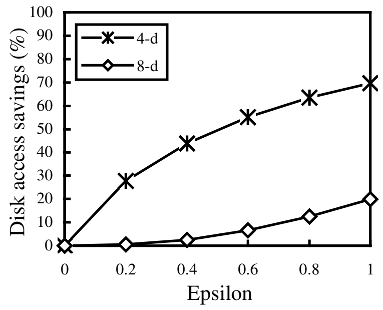


Fig. 7. Disk access savings

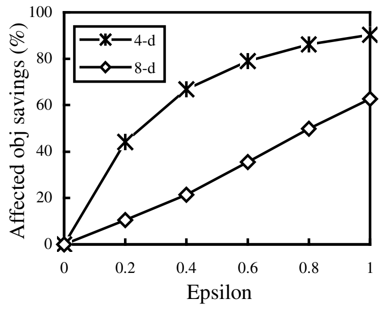


Fig. 8. Affected object savings

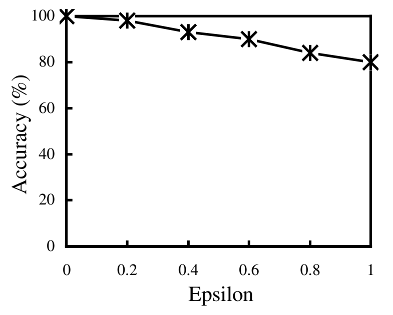


Fig. 9. Accuracy against the ε values

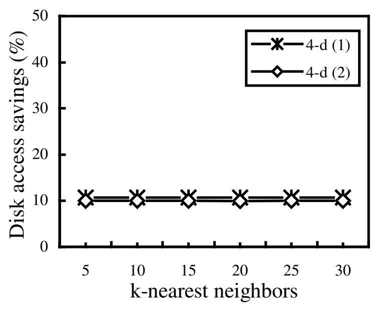


Fig. 10. Disk access savings

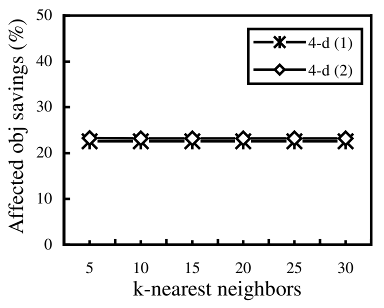


Fig. 11. Affected object savings

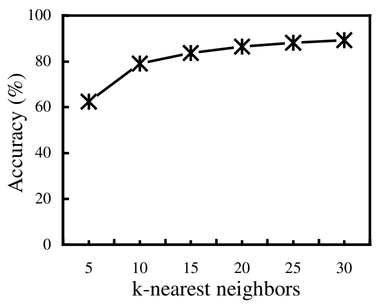


Fig. 12. Accuracy against the true k-NNs

The second experiment was done with two data sets of 4 dimensions. The ε value was set to 0.2. Here we did the experiment to show the efficiency of the ε-ISA in dealing with approximate k-nearest neighbor problem. The values of k were set to 5, 10, 15, 20, 25 and 30. Figures 10, 11 and 12 show the experimental results with notations similar to that of the previous experiment. The results obtained are quite good for small values of k and better for larger values of k. Specially, for k=25 or 30, disk access savings are over 10% and affected object savings are over 23% for each 4-dimensional feature space, while the accuracy is maintained at nearly 90%.

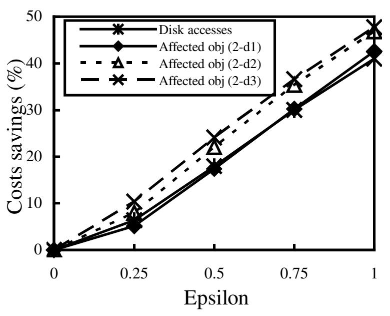


Fig. 13. Savings of Costs

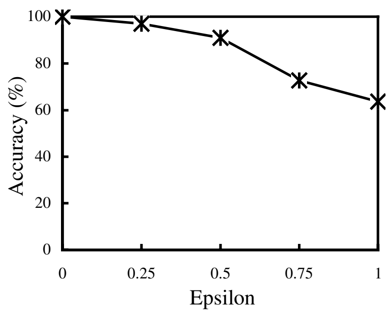


Fig. 14. Accuracy against the ε values

Our last experiment was performed on the data sets of 2 dimensions. For this experiment the ε-ISA was carried out to find (1+ε)-approximate NN for 3-condition queries. The main purpose of this test is to show effect of the query condition number on the efficiency of the ε-ISA. The ε values chosen were 0.25, 0.5, 0.75 and 1. The results are shown in figure 13 and 14: Figure 13 depicts all costs savings and figure 14 shows the accuracy, compared to the true NN of the selected queries, with respect to the ε values. The experimental results are particularly considerable. For ε=0.5, which means the tolerant error is permitted up to 50%, the ε-ISA saves nearly 18% of disk accesses and over 17%, 22% and 24% of the affected objects for each feature space, individually, while still preserving the accuracy of the approximate results at 91% with respect to the "true" NNs.

5 Conclusions and Future Work

In this paper we introduced an approach called ε-ISA to efficiently solving approximate nearest neighbor problem for complex vague queries. The ε-ISA becomes one of a few vanguard solutions for dealing with this problem, i.e. approximate M-FNN queries. Although the ε-ISA is presented in the context of the Vague Query System (VQS) [22], it can also be applied to a variety of application domains such as multimedia databases, GIS and tourist information systems, digital library systems, etc. In general the ε-ISA is very useful for application domains that the returned results need not to be exact but similar or approximately similar (with a certain tolerant error) to a given query. The experimental results have proven this. With a suitable ε value, the ε-ISA can save a very high percentage of the costs including both IO-cost and CPU-cost while it still preserves the accuracy of the returned results at a particularly very high value. Specially, the ε-ISA is applicable to general feature spaces including the ones that do not satisfy three conditions (see [31]) to make Fagin's algorithm [14] and all its variants as [26, 17] become utilizable.

Our future work will concentrate on deploying the ε-ISA for the real world applications as GIS and tourist information systems [29], addressing complex/multi-feature vague join operations, which must also deal with several (≥ 2) multidimensional data sets participating in the join operation [23].

References

1. S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. Journal of the ACM (JACM), November 1998, 45(6)
2. R. Baeza-Yates, B. Ribeiro-Neto. Modern Information Retrieval. ACM Press Books, 1999
3. C. Böhm, S. Berchtold, D. A. Keim. Searching in High-Dimensional Spaces. ACM Computing Surveys (CSUR), 33(3), September 2001
4. B. Braunmüller, M. Ester, H.P. Kriegel, J. Sander. Efficiently Supporting Multiple Similarity Queries for Mining in Metric Databases. ICDE 2000
5. S. Berchtold, H.P. Kriegel. S3: Similarity Search in CAD Database Systems. ACM SIGMOD International Conference on Management of Data, 1997

6. K.S. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft. When Is "Nearest Neighbor" Meaningful?. ICDT 1999
7. K. Böhm, M. Mlivoncic, H-J. Schek, R. Weber: Fast Evaluation Techniques for Complex Similarity Queries. VLDB 2001
8. S. Chaudhuri, L. Gravano. Evaluating Top-k Selection Queries. VLDB 1999
9. P. Ciaccia, M. Patella. PAC Nearest Neighbor Queries: Approximate and Controlled Search in High-Dimensional and Metric Spaces. ICDE 2000
10. T.K. Dang, J. Küng, R. Wagner. The SH-tree: A Super Hybrid Index Structure for Multidimensional Data. DEXA01, Springer Verlag, LNCS 2113, pp. 340-349, Munich, Germany, September 3-7, 2001
11. T.K. Dang, J. Küng, R. Wagner. Efficient Processing of k-Nearest Neighbor Queries in Spatial Databases with the SH-tree. Proc. of the 3rd iiWAS, September 10-12, Linz, Austria 2001
12. T. K. Dang, J. Küng, R. Wagner. ISA-An Incremental Hyper-Sphere Approach for Efficiently Solving Complex Vague Queries. DEXA 2002 (to appear)
13. T. K. Dang, J. Küng, R. Wagner. A General and Efficient Approach for Solving Nearest Neighbor Problem in the Vague Query System. The 3rd International Conference on Web-Age Information Management (WAIM), Beijing, August 11 - 13, 2002 (to appear)
14. R. Fagin. Combining Fuzzy Information from Multiple Systems. ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Montreal, Canada, 3-5 June 1996
15. C. Faloutsos, M. Ranganathan, Y. Manolopoulos. Fast subsequence matching in time-series databases. ACM SIGMOD International Conference on Management of Data, 1994
16. V. Gaede, O. Günther. Multidimensional Access Methods. ACM Computing Surveys (CSUR), 30(2), June 1998
17. U. Guentzer, W.T. Balke, W. Kiessling. Optimizing Multi-Feature Queries for Image Databases. VLDB 2000
18. G. R. Hjaltason and H. Samet. Ranking in Spatial Databases. Advances in Spatial Databases - 4th Symposium, SSD'95, LNCS 951, Springer-Verlag, Berlin, pp. 83-95, 1995
19. T. Ichikawa, M. Hirakawa. ARES: A Relational Database with the Capability of Performing Flexible Interpretation of Queries. IEEE Trans. on Software Engineering, Vol. 12, No. 5, 1986
20. J.M. Kleinberg. Two Algorithms for Nearest-Neighbor Search in High Dimensions. Annual ACM Symposium on Theory of Computing, 1997
21. E. Kushilevitz, R. Ostrovsky, Y. Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. Annual ACM Symposium on Theory of Computing, May 23-26, 1998, Dallas, Texas, USA
22. J. Kueng, J. Palkoska. VQS-A Vague Query System Prototype. IEEE Computer Society Press, DEXA 1997
23. J. Küng, J. Palkoska. Vague Joins-An Extension of the Vaque Query System VQS, IEEE Computer Society Press, DEXA 1998
24. J. Kueng, J. Palkoska. An Incremental Hypercube Approach for Finding Best Matches for Vague Queries. Springer-Verlag Berlin Heidelberg New York, Florenz, DEXA 1999
25. A. Motro. VAGUE: A User Interface to Relational Database that Permits Vague Queries. ACM Trans. on Office Information Systems, Vol 6, No. 3, July 1988
26. S. Nepal, M.V. Ramakrishna. Query Processing Issues in Image (Multimedia) Databases. ICDE 1999
27. M. Ortega, Y. Rui, K. Chakrabarti, S. Mehrotra, T.S. Huang. Supporting Similarity Queries in MARS. ACM International Conference on Multimedia, 1997
28. S. Pramanik, S. Alexander, J. Li. An Efficient Searching Algorithm for Approximate Nearest Neighbor Queries in High Dimensions. IEEE International Conference on Multimedia Computing and Systems, Vol. I, 7-11 June 1999, Florence, Italy

29. J. Palkoska, F. Pühretmair, A.M. Tjoa, R. Wagner, W. Wöß. Advanced Query Mechanisms in Tourism Information Systems. 9th International Conference on Information and Communication Technologies in Tourism, ENTER 2002
30. T. Seidl, H-P. Kriegel. Efficient User-Adaptable Similarity Search in Large Multimedia Databases. VLDB 1997
31. E.L. Wimmers, L.M. Haas, M.T. Roth, C. Braendli. Using Fagin's Algorithm for Merging Ranked Results in Multimedia Middleware. COOPIS 1999
32. R. Weber, H.J. Schek, S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. VLDB 1998
33. R. Fagin, A. Lotem, M. Naor. Optimal Aggregation Algorithms for Middleware. PODS 2001.
34. A. Natsev, Y.C. Chang, J.R. Smith, C.S. Li, J. S. Vitter. Supporting Incremental Join Queries on Ranked Inputs. VLDB 2001

Association Rule Extraction for Text Mining

M. Delgado, M.J. Martín-Bautista, D. Sánchez, J.M. Serrano, and M.A. Vila

Dpt. Computer Science and Artificial Intelligence, University of Granada
C/Periodista Daniel Saucedo s/n, 18071, Granada, Spain
{mdelgado, mbautis, daniel, jmserrano, vila@decsai.ugr.es}

Abstract. We present the definition of fuzzy association rules and fuzzy transactions in a text framework. The traditional mining techniques are applied to documents to extract rules. The fuzzy framework allows us to deal with a fuzzy extended Boolean model. Text mining with fuzzy association rules is applied to one of the classical problems in Information Retrieval: query refinement. The extracted rules help users to query the system by showing them a list of candidate terms to refine the query. Different procedures to apply these rules in an automatic and semi-automatic way are also presented.

1 Introduction

In general terms, the application of Data Mining and Knowledge Discovery techniques to text has been called Text Mining and Knowledge Discovery in Texts, respectively. The main difference to apply these techniques in a text framework is the special characteristics of text as unstructured data, totally different from databases, where mining techniques are usually applied and structured data is managed. Some general approaches about Text Mining and Knowledge Discovery in Texts can be found in [10], [13], [17], [18].

Traditional mining techniques deal with Boolean association rules in the sense that they are generated from a set of Boolean transactions representing that an attribute is present in the transaction by 1 and non present by 0. However, most of the data implies the handling of uncertainty and quantitative attributes such as the age or the weight. The theory of fuzzy sets has been revealed as a good tool to deal with this kind of data in traditional data mining in relational databases [8], [9], [14]. In general, fuzzy association rules and fuzzy transactions can be defined to deal with structure data such as in relational databases and unstructured data, such as texts.

In this work, fuzzy association rules applying techniques from data mining will be discovered in a text framework as a process to select terms to be added to an original query. Some other approaches can be found in this direction. In [25] a vocabulary generated by association rules is used to improve the query. In [12] a system for Finding Associations in Collections of Text (FACT) is presented. The system takes background knowledge to show the user a simple graphical interface providing a query language with well-defined semantics for the discovery actions based on term taxonomy at different granularity levels. A different application of association rules but in the Information Retrieval framework can be found in [20] where the extracted rules are employed for document classification.

This paper is organized as follows: from section 2 to section 4, general theory about association rules, fuzzy association rules and fuzzy transactions is presented. In section 5, an application to text framework is proposed and a definition of text transactions is given. In section 6, the extracted text association rules are applied to query reformulation in an Information Retrieval framework. Finally, conclusions and future trends are given in section 7.

2 Association Rules

The obtaining and mining of association rules is one of the main research problems in data mining framework [1]. Given a database of transactions, where each transaction is an itemset, the obtaining of association rules is a process guided by the constrains of *support* and *confidence* specified by the user. Support is the percentage of transactions containing an itemset, calculated in a statistical manner, while confidence measures the strength of the rule. Formally, let *T* be a set of transactions containing items of a set of items *I*. Let us consider two itemsets $I_1, I_2 \subseteq I$, where $I_1 \cap I_2 = \emptyset$. A rule $I_1 \Rightarrow I_2$ is an implication rule meaning that the apparition of itemset I_1 implies the apparition of itemset I_2 in the set of transactions *T*. I_1 and I_2 are called antecedent and consequent of the rule, respectively.

Given a support of an itemset noted by $supp(I_k)$, and the rule $I_1 \Rightarrow I_2$, the support and the confidence of the rule noted by $Supp(I_1 \Rightarrow I_2)$ and $Conf(I_1 \Rightarrow I_2)$, respectively, are calculated as follows:

$$Supp(I_1 \Rightarrow I_2) = supp(I_1 \cup I_2) \quad \textbf{(1)}$$

$$Conf(I_1 \Rightarrow I_2) = \frac{supp(I_1 \cup I_2)}{supp(I_1)} \quad \textbf{(2)}$$

The constrains of minimum support and minimum confidence are established by the user with two threshold values: *minsupp* for the support and *minconf* for the confidence. A *strong rule* is an association rule whose support and confidence are greater that thresholds minsupp and minconf, respectively. Once the user has determined these values, the process of obtaining association rules can be decomposed in two different steps:

1. Find all the itemsets that have a support above threshold *minsupp*. These itemsets are called *frequent itemsets*.
2. Generate the rules, discarding those rules below threshold *minconf*.

The rules obtained with this process are called boolean association rules in the sense that they are generated from a set of boolean transactions where the values of the tuples are 1 or 0 meaning that the attribute is present in the transaction or not, respectively. However, the consideration of rules coming from real world implies, most of the times, the handling of uncertainty and quantitative association rules, that is, rules with quantitative attributes such as, for example, the age or the weight of a person. To solve this, one of the solutions is to make intervals considering all the possible values that can take the quantitative attributes [21]. The other solution to this problem is the use the theory of fuzzy sets to model quantitative data and, therefore, deal with the problem of quantitative rules. The rules generated using this theory are called fuzzy association rules, and their principal bases as well as the concept of fuzzy transactions are presented in next section [9], [14].

3 A Fuzzy Framework for Association Rules

Fuzzy association rules are defined as those rules that associate items of the form *(Attribute, Label)*, where the label has an internal representation as fuzzy set over the domain of the attribute [5]. The obtaining of these rules comes from the consideration of fuzzy transactions. In the following, we present the main and features related to fuzzy transactions and fuzzy association rules. The complete model and applications of these concepts can be found in [9].

3.1 Fuzzy Transactions

Given a finite set of items *I*, we define a fuzzy transaction as any nonempty fuzzy subset $\tilde{\tau} \subseteq I$. For every $i \in I$, the membership degree of *i* in a fuzzy transaction $\tilde{\tau}$ is noted by $\tilde{\tau}(i)$. Therefore, given an itemset $I_o \subseteq I$, we note $\tilde{\tau}(I_0)$ the membership degree of I_0 to a fuzzy transaction $\tilde{\tau}$. We can deduce from this definition that boolean transactions are a special case of fuzzy transactions. We call FT-set the set of fuzzy transactions, remarking that it is a crisp set.

A set of fuzzy transactions FT-set is represented as a table where columns and rows are labeled with identifiers of items and transactions, respectively. Each cell of a pair *(transaction, itemset)* of the form $(I_0, \tilde{\tau}_j)$ contains the membership degree of I_o in $\tilde{\tau}_j$, noted $\tilde{\tau}_j(I_0)$ and defined as:

$$\tilde{\tau}(I_0) = \min_{i \in I_0} \tilde{\tau}(i) \tag{3}$$

The representation of an item I_o in a FT-set *T* based in *I* is represented by a fuzzy set $\tilde{\Gamma}_{I_0} \subseteq T$, defined as

$$\tilde{\Gamma}_{I_0} = \sum_{\tilde{\tau} \in T} \tilde{\tau}(I_0)/\tilde{\tau} \tag{4}$$

3.2 Fuzzy Association Rules

A fuzzy association rule is a link of the form $A \Rightarrow B$ such that $A, B \subset I$ and $A \cap B = \varnothing$, where *A* is the antecedent and *B* is the consequent of the rule, being both of them fuzzy itemsets. An ordinary association rule is a fuzzy association rule. The meaning of a fuzzy association rule is, therefore, analogous to the one of an ordinary association rule, but the set of transactions where the rule holds, which is a FT-set. If we call $\tilde{\Gamma}_A$ and $\tilde{\Gamma}_B$ the degrees of attributes *A* and *B* in every transaction $\tilde{\tau} \in T$, we can assert that the rule $A \Rightarrow B$ holds with totally accuracy in *T* when $\tilde{\Gamma}_A \subseteq \tilde{\Gamma}_B$. For more details about the definition of fuzzy association rules, see [9].

4 Measures for Fuzzy Association Rules

The imprecision latent in fuzzy transactions makes us consider a generalization of classical measures of support and confidence by using approximate reasoning tools. One of these tools is the evaluation of quantified sentences presented in [26]. A quantified sentence is and expression of the form "*Q* of *F* are *G*", where *F* and *G* are two fuzzy subsets on a finite set *X*, and *Q* is a relative fuzzy quantifier. We focus on quantifiers representing fuzzy percentages with fuzzy values in the interval [0,1] such as "most", "almost all" or "many". These quantifiers are called *relative quantifiers.*

Let us consider Q_M a quantifier defined as $Q_M(x) = x, \forall x \in [0,1]$. We define the *support of an itemset* I_0 in an FT-set *T* as the evaluation of the quantified sentence Q_M *of T are* $\tilde{\Gamma}_{I_0}$ while the *support of a rule* $A \Rightarrow B$ in *T* is given by the evaluation of Q_M *of T are* $\tilde{\Gamma}_{A \cup B} = Q_M$ *of T are* $\tilde{\Gamma}_A \cap \tilde{\Gamma}_B$ and its confidence is the evaluation of Q_M *of* $\tilde{\Gamma}_A$ *are* $\tilde{\Gamma}_B$.

We evaluate the sentences by means of method GD presented in [6]. To evaluate the sentence "*Q* of *F* are *G*", a compatibility degree between the relative cardinality of G with respect to F and the quantifier is represented by $GD_Q(G/F)$ and defined as

$$Q\left(\frac{|F \cap G|}{|F|}\right) \tag{5}$$

when *F* and *G* are crisp. The GD method verifies this property. For more details, see [6]. We can interpret the ordinary measures of confidence and support as the degree to which the confidence and support of an association rule is Q_M. Other properties of this quantifier can be seen in [9].

This generalization of the ordinary measures allows us, using Q_M, provide an accomplishment degree, basically. Hence, for fuzzy association rules we can assert

$$Q_M \tau \in T, A \Rightarrow B \tag{6}$$

4.1 Measuring Accuracy of a Fuzzy Rule by Certainty

We propose the use of certainty factors to measure the accuracy of association rules. A previous study can be found in [7].

We define the certainty factor (*CF*) of a fuzzy association rule $A \Rightarrow B$ based on the value of the confidence of the rule. If $Conf(A \Rightarrow B) > supp(B)$ the value of the factor is given by expression (7); otherwise, is given by expression (8), considering that if *supp(B)*=1, then $CF(A \Rightarrow B) = 1$ and if *supp(B)*=0, then $CF(A \Rightarrow B) = -1$

$$CF(A \Rightarrow B) = \frac{Conf(A \Rightarrow B) - supp(B)}{1 - supp(B)} \tag{7}$$

$$CF(A \Rightarrow B) = \frac{Conf(A \Rightarrow B) - supp(B)}{supp(B)} \tag{8}$$

We demonstrated in [2] that certainty factors verify the three properties by [21]. From now on, we shall use certainty factors to measure the accuracy of a fuzzy association rule. We consider a fuzzy association rule as strong when its support and certainty factor are greater than thresholds *minsupp* and *minCF*, respectively.

5 Text Mining

The main problem when the general techniques of data mining are applied to text is to deal with unstructured data, in comparison to structured data coming from relational databases. Therefore, with the purpose to perform a knowledge discovery process, we need to obtain some kind of structure in the texts. Different representations of text for association rules extraction have been considered: bag of words, indexing keywords, term taxonomy and multi-term text phrases [10]. In our case, we use automatic indexing techniques coming from Information Retrieval [23]. We represent each document by a set of terms with a weight meaning the presence of the term in the document. Some weighting schemes for this purpose can be found in [24]. One of the more successful and more used representation schemes is the *tf-idf* scheme, which takes into account the term frequency and the inverse document frequency, that is, if a term occurs frequently in a document but infrequently in the collection, a high weight will be assigned to that term in the document. This is the scheme we consider in this work. The algorithm to get the representation by terms and weights of a document d_i can be detailed by the known following the steps detailed below.

1. Let $D = \{d_1, \ldots d_n\}$ be a collection of documents
2. Extract an initial set of terms *S* from each document $d_i \in D$
3. Remove stop words
4. Apply stemming (*via* Porter's algorithm [22])
5. The representation of d_i obtained is a set of keywords $\{t_1, \ldots, t_m\} \in S$ with their associated weights $\{w_1, \ldots, w_m\}$

5.1 Text Transactions

From a collection of documents $D=\{d_1,\ldots,d_n\}$ we can obtain a set of terms $I=\{t_1,\ldots,t_m\}$ which is the union of the keywords for all the documents in the collection, obtained from the algorithm included in the previous section. The weights associated to these terms are represented by $W=\{w_1,\ldots,w_m\}$. Therefore, for each document d_i, we consider an extended representation where a weight of 0 will be assigned to every term appearing in some of the documents of the collection but not in d_i.

Considering these elements, we can define a *text transaction* $\tau_i \in T$ as the extended representation of document d_i. Without loosing generalization, we can write $T=\{d_1,\ldots,d_n\}$. However, as we are dealing with fuzzy association rules, we will consider a fuzzy representation of the presence of the terms in documents, by using the normalized tf-idf scheme [19]. Analogously to the former case, we can define a set of *fuzzy text transactions* $FT=\{d_1,\ldots,d_n\}$, where each document d_i corresponds to a fuzzy transaction $\tilde{\tau}_i$, and where the weights $W=\{w_1,\ldots,w_m\}$ of the keyword set $I=\{t_1,\ldots,t_m\}$ are fuzzy values.

6 Text Mining for Query Refinement

When a user try to express his/her needs in a query, the terms that finally appear in the query are usually not very specific due to the lack of background knowledge of the user about the topic or just because in the moment of the query, the terms do not come to the user's mind. To help the user with the query construction, terms related to the words of a first query may be added to the query.

From a first set of documents retrieved, data mining techniques are applied in order to find association rules among the terms in the set. The more accurate rules that include the original query words in the antecedent / consequent of the rule, are used to modify the query by automatically adding these terms to the query or, by showing to the user the related terms in those rules, so the modification of the query depends on the user's decision. A generalization or specification of the query will occur when the terms used to reformulate the query appear in the consequent / antecedent of the rule, respectively. This suggestion of terms helps the user to reduce the set of documents, leading the search through the desired direction.

This process can be understood as a feedback from the user part, but a term level, not at document level as traditionally is performed [4]. In this way, the relevance feedback is available during the query expansion, since the terms added to the query are directly selected by the user from a list generated by the system in a semi-automatic query refinement process, which is detailed in section 6.2. This process differs from the traditional relevance feedback methods, where the relevance is assigned to some retrieved documents by the user, and the terms to be added to the query are selected from the top-ranked documents retrieved once the feedback of the user has been incorporated to the original rank given by the system [16].

This problem of query optimization has been broadly treated in the field of Information Retrieval. We can find several references with solutions to this problem. A good review of the topic can be found in [11] and [15].

6.1 Generalization and Specialization of a Query

Once the first query is constructed, and the association rules are extracted, we make a selection of rules where the terms of the original query appear. However, the terms of the query can appear in the antecedent or in the consequent of the rule. If a query term appears in the antecedent of a rule, and we consider the terms appearing in the consequent of the rule to expand the query, a generalization of the query will be carried out. Therefore, a generalization of a query gives us a query on the same topic as the original one, but looking for more general information.

However, if query term appears in the consequent of the rule, and we reformulate the query by adding the terms appearing in the antecedent of the rule, then a specialization of the query will be performed, and the precision of the system should increase. The specialization of a query looks for more specific information than the original query but in the same topic. In order to obtain as much documents as possible, terms appearing in both sides of the rules can also be considered.

6.2 Query Refinement Procedure

By means of fuzzy association rules, we can provide a system with a query reformulation ability in order to improve the retrieval process. We represent a query $Q = \{q_1, \ldots, q_k\}$ with associated weights $P = \{p_1, \ldots, p_k\}$. To obtain a relevance value for each document, the query representation is matched to each document representation, obtained as explained in section 5. If a document term does not appear in the query, its value will be assumed as 0. The considered operators and measures are the one from the generalized Boolean model with fuzzy logic [3].

The user's initial query generates a set of ranked documents. If the top-ranked documents do not satisfy user's needs, the query improvement process starts. From the retrieved set of documents, association relations are found. At this point, two different approaches can be considered: an automatic expansion of the query or a semi-automatic expansion, based on the intervention of the user in the selection process of the terms to be added to the query. The complete process in both cases are detailed in the following:

Automatic query refinement process

1. The user queries the system
2. A first set of documents is retrieved
3. From this set, the representation of documents is extracted and fuzzy association rules are generated.
4. The terms co-occurring in the rules with the query terms are added to the query, depending on a generalization or specialization process.
5. With the expanded query, the system is queried again.

Semi-automatic query refinement process

1. The user queries the system
2. A first set of documents is retrieved
3. From this set, the representation of documents and fuzzy association rules are generated
4. The terms co-occurring in the rules with the query terms are shown to the user, following a generalization or specialization process
5. The user selects those terms more related to her/his needs
6. The selected terms are added to the query, which is used again to query the system

7 Concluding Remarks and Future Work

In this work, the application of traditional data mining techniques to text has been presented. The handling of uncertainty and quantitative attributes by means of fuzzy sets has been defined by the concepts of fuzzy transactions and fuzzy association rules. The application of these definitions in documents has been presented by the identification of with fuzzy transactions and fuzzy association rules with relations among terms presented in the document collection. The fuzzy framework in the mining environment allow us to work with a weighted extended Boolean model with fuzzy logic in the document and query space. Finally, an application of these elements to the problem of query optimization in Information Retrieval has been presented, identifying the generalization and specialization of the query with the inclusion of terms appearing in the antecedent and in the consequent of the rule, respectively.

In the present, we are experimenting with the presented approach and application to query refinement. A comparison to other classical approaches coming from Information Retrieval to solve the query optimization problem will be performed in the future.

References

1. Agrawal, R., Imielinski, T., Swami, A. Mining Association Rules between Set of Items in Large Databases. *Proc. of the 1993 ACM SIGMOD Conference*, pp. 207-216, 1993.
2. Berzal, F., Delgado, M., Sánchez, D., Vila, M.A. Measuring the accuracy and importance of association rules. Tech. Rep. CCIA-00-01-16, Department of Computer Science and Artificial Intelligence, University of Granada, 2000.
3. Buell, D.A., Kraft, D.H. Performance Measurement in a Fuzzy Retrieval Environment. In *Proceedings of the Fourth International Conference on Information Storage and Retrieval, ACM/SIGIR Forum*, 16(1), pp. 56-62, Oakland, CA, 1981.
4. Chang, C.H., Hsu, C.C. Enabling Concept-Based Relevance Feedback for Information Retrieval on the WWW. *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no.4, 1999.
5. Delgado, M., Sánchez, D., Vila, M.A. Acquisition of fuzzy association rules from medical data. In Barro, S. and Marín, R. (Eds.) *Fuzzy Logic in Medicine,* Physica-Verlag, 2000.
6. Delgado, M., Sánchez, D., Vila, M.A. Fuzzy cardinality based evaluation of quantified sentences. *International Journal of Approximate Reasoning,* vol. 23, pp. 23-66, 2000.

7. Delgado, M., Martín-Bautista, M.J., Sánchez, D., Vila, M.A. Mining strong approximate dependences from relational databases. *Proc. Of IPMU* 2000, Madrid.
8. Delgado, M., Martín-Bautista, M.J., Sánchez, D., Vila, M.A. Mining association rules with improved semantics in medical databases. *Artificial Intelligence in Medicine* vol. 21, pp. 241-245, 2001.
9. Delgado, M., Marín, N., Sánchez, D., Vila, M.A. Fuzzy Association Rules: General Model and Applications. *IEEE Transactions of Fuzzy Systems*, vol. 126, no.2, pp. 41-54, 2002.
10. Delgado, M., Martín-Bautista, M.J., Sánchez, D., Vila, M.A. Mining Text Data: Special Features and Patterns. *Proc. of EPS Exploratory Workshop on Pattern Detection and Discovery in Data Mining, Imperial College London, UK, September 2002.*
11. Efthimiadis, R. Query Expansion. *Annual Review of Information Systems and Technology,* vol. 31, pp. 121-187, 1996.
12. Feldman, R., Hirsh, H. Mining associations in text in the presence of Background Knowledge *Proc. of the Second International Conference on Knowledge Discovery from Databases,* 1996.
13. Feldman, R., Fresko, M., Kinar, Y., Lindell, Y., Liphstat, O., Rajman, M., Schler, Y., Zamir, O. Text Mining at the Term Level. *Proc. of the 2nd European Symposium of Principles of Data Mining and Knowledge Discovery*, pp. 65-73, 1998.
14. Fu, A.W., Wong, M.H., Sze, S.C., Wong, W.C., Wong, W.L., Yu, W.K. Finding Fuzzy Sets for the Mining of Fuzzy Association Rules for Numerical Attributes, *Proc. of Int. Symp. on Intelligent Data Engineering and Learning* (IDEAL'98), Hong Kong, pp.263-268, 1998.
15. Gauch, S., Smith, J.B. An Expert System for Automatic Query Reformulation. *Journal of the American Society for Information Science*, 44(3), pp. 124-136.
16. Harman, D.K. "Relevance Feedback and Other Query Modification Techniques". In W.B. Frakes and R. Baeza-Yates (Eds.) *Information Retrieval: Data Structures and Algorithms*, pp. 241-263, Prentice Hall, 1992.
17. Hearst, M. Untangling Text Data Mining. *Proc. of the 37th Annul Meeting of the Association for Computational Linguistics (ACL'99),* University of Maryland, June 1999.
18. Kodratoff, Y. Knowledge Discovery in Texts: A Definition, and Applications. In Z. W. Ras and A. Skowron (Eds.) *Foundation of Intelligent Systems*, Lectures Notes on Artificial Intelligence 1609, Springer Verlag, 1999.
19. Kraft, D.H., Petry, F.E., Buckles, B.P., Sadasivan, T. Genetic Algorithms for Query Optimization in Information Retrieval: Relevance Feedback. In E. Sanchez, T. Shibata and L. Zadeh, (Eds.), *Genetic Algorithms and Fuzzy Logic Systems*, in Advances in Fuzziness: Applications and Theory, vol. 7, pp. 157-173, World Scientific.
20. Lin, S.H., Shih, C.S., Chen, M.C., Ho, J.M., Ko, M.T., Huang, Y.M. Extracting Classification Knowledge of Internet Documents with Mining Term Associations: A Semantic Approach. *Proc. of ACM/SIGIR'98,* pp. 241-249, Melbourne, Australia, 1998.
21. Piatetsky-Shapiro, G. Discovery, Analysis, and Presentation of Strong Rules. In Piatetsky-Shapiro, G. and Frawley W.J. (Eds.) *Knowledge Discovery in Databases*, AAAI/MIT Press, 1991.
22. Porter, M.F. An algorithm for suffix stripping. *Program,* 14(3): 130-137, 1980.
23. Salton, G., McGill, M.J. Introduction to Modern Information Retrieval. McGraw-Hill, 1983.
24. Salton, G., Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management,* vol. 24, no. 5, pp. 513-523, 1988.
25. Srinivasan, P., Ruiz, M.E., Kraft, D.H., Chen, J. Vocabulary mining for information retrieval: rough sets and fuzzy sets. *Information Processing and Management*, 37, pp. 15-38, 2001.
26. Zadeh, L.A. A computational approach to fuzzy quantifiers in natural languages. *Computing and Mathematics with Applications,* vol. 9, no. 1, pp. 149-184, 1983.

User Preference Information in Query Answering[⋆]

Pierangelo Dell'Acqua[1,2], Luís Moniz Pereira[2], and Aida Vitória[1]

[1] Department of Science and Technology - ITN
Linköping University, 601 74 Norrköping, Sweden
{pier,aidvi}@itn.liu.se
[2] Centro de Inteligência Artificial - CENTRIA
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
lmp@di.fct.unl.pt

Abstract. This paper discusses the use of usage information to enhance cooperative behaviour from query answering systems. A user can pose a query to the system by providing (at query time) update and preference information. Updates allow us to model dynamically evolving worlds and preferences allow us to facilitate the retrieval of information by targeting the answers of the system with respect to users' interests in a given context.

1 Introduction

In the field of cooperative answering, Minker [9] surveys foundational work done toward developing query answering systems with the ability to exhibit cooperative behaviour. He argues that query answering systems are often difficult to use because they do not attempt to cooperate with their users, and he advocates among other things the use of additional information about the users, e.g., their goals, interests and preferences.

In query answering systems, queries have typically several answers and selecting the desired ones is often hard for users. In many applications it is desirable to enhance the system with preference information that can be employed to derive the intended conclusions. Consider a system whose knowledge is defined as a pair (P, R), where P is a set of rules formalizing the background knowledge of the system and R expresses preference information over the rules in P, meaning that when rules conflict and exclude one another, then some are preferred over others according to a partial order. Now the system can use the knowledge in P to derive conclusions and the preferences in R to derive the preferred ones.

The system will exhibit an extra level of flexibility if further it permits the user to provide a temporary preference information update at query time, contextual to some goal. A query can be defined as (G, Pref), where the idea is to

[⋆] Pereira acknowledges the support of POCTI project 40958 "FLUX - FleXible Logical Updates" and Dell'Acqua a Research Grant by the *Stiftelsen Lars Hiertas Minne.*

prove the goal G by taking into account the preferences in *Pref*. Now, if the knowledge of the system is (P, R), then the system has to derive the conclusions that satisfy the preferences in R (i.e., the preferences applying to every query) but which are updated by the contextual preferences in *Pref* (i.e, the preferences specific to that particular query). Finally, it is desirable to make the knowledge (P, R) of the system updatable in a way that it can be modified to reflect changes in the world, including the general preferences.

In this paper, we elaborate on the usage of user information in cooperative answering, as advocated by Minker. We present a novel framework that allows the user to specify his/her preferences and context information. Both user preferences and context information are used for building an answer. Context information provided by a query is taken into account through the mechanism of updates. Updates are used to model dynamically evolving worlds and preferences to enhance the retrieval of information. Given a piece of knowledge describing the world, and given a change in the world, the problem is how to modify the knowledge to cope with that change. The key issue is how to accommodate, in the represented knowledge, any changes in the world. In this setting it may well happen that changes in the world contradict previous knowledge, i.e. the union of the previous knowledge with the representation of the new knowledge has no model. It is up to updates to remove from the prior knowledge representation any piece of knowledge that changed, and to replace it with the newer one.

Preference reasoning is a pervasive form of reasoning. Preference information is used along with incomplete knowledge. In such a setting, due to the incompleteness of the knowledge, several semantic models may be possible. Preference reasoning acts by choosing among those possible models. An example is the "vacation" problem where the incomplete knowledge contains the rules: "go to the beach if not go to the mountain" (1) and "go to the mountain if not go to the beach" (2). Then, two models are possible: one model obtained by preferring the first rule, and the other obtained by preferring the second rule. Thus, preference information among the rules can be used to select some of the models on the basis of any chosen criteria expressed by the user (e.g., utility, plausibility, etc.). Cf. [4,5] for additional motivation, comparisons, applications, and references. To the best of our knowledge the approach in [1,3] is the only approach proposed so far that combines update and preference reasoning, the updating of preferences included.

2 Logic Programming Framework

We consider propositional Horn theories. In particular, we represent default negation $not\, A$ as standard propositional variable (atom). Suppose that $\mathcal{K}$ is an arbitrary set of propositional variables whose names do not begin with a "not". By the propositional language $\mathcal{L}_{\mathcal{K}}$ generated by $\mathcal{K}$ we mean the language whose set of propositional variables consists of $\{A, not\, A : A \in \mathcal{K}\}$. Atoms $A \in \mathcal{K}$ are called *objective atoms* while the atoms $not\, A$ are called *default atoms*. From the definition it follows that the two sets are disjoint. Objective and default atoms are generically called *literals*.

Definition 1 (Generalized logic program). *A generalized logic program over the language $\mathcal{L}_{\mathcal{K}}$ is a set of ground generalized rules of the form $L_0 \leftarrow L_1, \ldots, L_n$ ($n \geq 0$), where each L_i is a literal over $\mathcal{L}_{\mathcal{K}}$.*

By $head(r)$ we mean L_0, by $body(r)$ the set of literals $\{L_1, \ldots, L_n\}$, by $body^+(r)$ the set of all objective atoms in $body(r)$, and by $body^-(r)$ the set of all default atoms in $body(r)$. Whenever a literal L is of the form $not\, A$, $not\, L$ stands for the objective atom A.

The semantics of generalized logic programs [7,8] is defined as a generalization of the stable model semantics [6]. Here we use the definition that appeared in [2] (proven there equivalent to [7,8]). In the remainder of the paper, by (2-valued) *interpretation* M of $\mathcal{L}_{\mathcal{K}}$ we mean any set of propositional variables from $\mathcal{L}_{\mathcal{K}}$ such that for any A in $\mathcal{L}_{\mathcal{K}}$ precisely one A or $not\ A$ belongs to M.

Definition 2 (Default assumptions). *Let P be a set of generalized rules and M an interpretation of P. Then*

$$Default(P, M) = \{not\, A \mid \not\exists r \in P : head(r) = A \text{ and } M \models body(r)\}.$$

Definition 3 (Stable Models of Generalized Programs). *Let P be a generalized logic program and M an interpretation of P. M is a (regular) stable model of P iff $M = least(P \cup Default(P, M))$.*

To express preference information in logic programs, we introduce the notion of priority rule and prioritized logic program. Let $N = \{n_{r_1}, \ldots, n_{r_k}\}$ be a name set containing a unique name for every generalized rule over the language $\mathcal{L}_{\mathcal{K}}$. Given a rule r, we write $n_r \in N$ to indicate its name. Let $<$ be a binary predicate symbol[1] whose arguments are names in N. $n_r < n_u$ means that rule r is preferred to rule u. Denote by $\bar{\bar{\mathcal{K}}}$ the following superset of the set $\mathcal{K}$ of propositional variables $\bar{\bar{\mathcal{K}}} = \mathcal{K} \cup \{n_r < n_u : \{n_r, n_u\} \subseteq N\}$.

Definition 4 (Priority rule). *A priority rule over the language $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$ is a generalized rule $Z \leftarrow L_1, \ldots, L_n$ with $n \geq 0$, where Z is a literal of the form $n_r{<}n_u$ or its default negation ($not\ n_r{<}n_u$) and each L_i is a literal over $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$.*

Note that the set N does not include $<$, and that $<$ can only occur in the head and (possibly) in the body of priority rules.

Definition 5 (Prioritized logic program). *Let P be a generalized logic program over the language $\mathcal{L}_{\mathcal{K}}$ and R a set of priority rules over the language $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$. Then $\Sigma = (P, R)$ is a prioritized logic program.*

The generalized rules in P formalize the domain knowledge while the priority rules in R express preferences among the rules in P.

Definition 6 (Goal). *Let $L_1, \ldots, L_n$ be literals over the language $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$. Then $?{-}L_1, \ldots, L_n$ is a goal.*

[1] In order to establish the preferred stable models (cf. Def. 12), we require the relation induced by $<$ to be a well-founded, strict partial ordering on program rules.

3 Dynamic Prioritized Programs

In this section we recall the approach of Dynamic Logic Programming [2] that can be used to model the evolution of prioritized logic programs through sequences of updates (including the update of priority rules). In Dynamic Logic Programming, sequences of generalized programs $P_1 \oplus \cdots \oplus P_s$ are given. Intuitively, a sequence may be viewed as the result of, starting with program P_1, updating it with program P_2, ..., and updating it with program P_s. In such a view, dynamic logic programs are to be used in knowledge bases that evolve. New rules (coming from newly acquired knowledge) can be added at the end of the sequence without bothering whether they conflict with previous knowledge. The role of Dynamic Logic Programming is to ensure that these newly added rules are in force, and that previous rules are still valid (by inertia) as far as possible, i.e. they are kept for as long as they do not conflict with newly added ones.

To allow the updating of priority rules, the notion of dynamic prioritized programs has been introduced into Dynamic Logic Programming. Instead of a sequence of programs representing knowledge, the idea is to use a sequence of pairs: of programs representing knowledge, and of sets of priority rules describing the priority relation among rules of the knowledge representation. In general, an update of the priority rules may depend on some other predicate. To permit this generality, priority rules are allowed to refer to predicates defined in the programs that represent knowledge. In the remainder, let $S = \{1, \dots, s, \dots\}$ be a set of natural numbers. We call the elements $i \in S$ *states*.

Definition 7 (Dynamic Prioritized Program). *Let S be a set of states. For every state $i \in S$, let P_i be the generalized logic program over the language $\mathcal{L}_{\mathcal{K}}$, and R_i the set of priority rules over the language $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$. Then, $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ is a dynamic prioritized program over the language $\mathcal{L}_{\bar{\bar{\mathcal{K}}}}$.*

The semantics of a dynamic prioritized program is defined according to the rationale above. Given a model M of the program P_s, start by removing all the rules from previous programs whose head is the complement of some later rule with true body in M (i.e. by removing all rules which conflict with more recent ones). All other persist through by inertia. Then, as for the stable models of a single generalized program, add facts *not* A for all atoms A which have no rule at all with true body in M, and compute the least model. If M is a fixpoint of this construction, M is a stable model of the sequence up to P_s.

Definition 8 (Rejected rules). *Let $s \in S$ be a state. Let $\Gamma = \bigoplus\{(P_i \cup R_i) : i \in S\}$ be a dynamic prioritized program and M an interpretation of Γ. Then*

$$\begin{aligned} &\mathit{Reject}(s, M, \Gamma) = \\ &\quad \{r \in T_i \mid \exists r' \in T_j, \mathit{head}(r) = \mathit{not}\, \mathit{head}(r'), i < j \leq s \text{ and } M \models \mathit{body}(r')\} \end{aligned}$$

where $T_i = P_i \cup R_i$, for every $i \in S$.

The stable models of a dynamic prioritized program Γ are defined in terms of the stable models of the dynamic logic program obtained by the union of all the P_is and R_is in Γ. Consequently, the preference information expressed by the priority rules in the R_is is not yet taken into account here. (In Section 4 we define the notion of preferred stable models that takes priority rules into consideration.)

Definition 9 (Stable Models of a DPP at state s). *Let $s \in S$ be a state. Let $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ be a dynamic prioritized program and M an interpretation of Γ. M is a stable model of Γ at state s iff*

$$M = least([\mathcal{PR} - Reject(s, M, \Gamma)] \cup Default(\mathcal{PR}, M))$$

where $\mathcal{PR} = \bigcup_{i \leq s}(P_i \cup R_i)$.

The next example illustrates the stable models of a dynamic prioritized program. Note that the preference information expressed in the R_is is not used to select the stable models.

Example 1. Let $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ be the dynamic prioritized program:

$$P_1 = \left\{ \begin{array}{ll} a \leftarrow not\, b & (r_1) \\ b \leftarrow not\, a & (r_2) \end{array} \right\} \qquad R_1 = \{\, n_1 < n_2 \,\}$$

$$P_2 = \{\, c \,\} \qquad\qquad R_2 = \{\, not\, (n_1 < n_2) \,\}$$

The stable models of Γ at state 1 are $M_1 = \{a, n_1 < n_2\}$ and $M_2 = \{b, n_1 < n_2\}$, and at state 2 are $M_3 = \{a, c\}$ and $M_4 = \{b, c\}$.

4 Preferred Stable Models

We present the notion of preferred stable model (for a more detailed presentation see [3]). Intuitively, the priority information specified by the R_is in a dynamic prioritized program Γ is used to prefer among the stable models of Γ. This is achieved by first deleting (from Γ) all the rejected rules according to updates and then by deleting all the rules that are unpreferred according to preference information. The set of unpreferred rules contains (1) the rules defeated by the head of some more preferred rule, (2) the rules that attack a more preferred rule, and (3) the unsupported rules.

Definition 10 (Unsupported rules). *Let P be a set of generalized rules and M an interpretation of P. Then*

$$Unsup(P, M) = \{r \in P : M \models head(r) \text{ and } M \not\models body^-(r)\}.$$

Definition 11 (Unpreferred generalized rule). *Let P be a set of generalized rules and M an interpretation of P. $Unpref(P, M)$ is a set of unpreferred generalized rules of P and M iff*

$$Unpref(P, M) = least(Unsup(P, M) \cup \mathcal{X})$$

where $\mathcal{X} = \{r \in P \mid \exists r' \in (P - Unpref(P, M))$ *such that:*
$M \models r' < r,\ M \models body^+(r')$ *and*
$[\ not\ \ head(r') \in body^-(r)$ *or* $(not\ \ head(r) \in body^-(r'),\ M \models body(r))\ ]\ \}$.

As the set N of constants of $<$ does not include $<$ itself, it is not possible to express preferences on priority rules themselves. Therefore, no priority rule r in P can be unpreferred, that is, $r \notin Unpref(P, M)$, for any set P of generalized rules and any interpretation M of P.

Definition 12 (Preferred Stable Models at state s). *Let $s \in S$ be a state. Let $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ be a dynamic prioritized program and M a stable model of Γ at state s. Then M is a preferred stable model of Γ at state s iff*

- $\forall r : (r < r) \notin M$
- $\forall r_1, r_2, r_3 :$ *if* $(r_1 < r_2) \in M$ *and* $(r_2 < r_3) \in M$ *then* $(r_1 < r_3) \in M$
- $M = least(\,[\,\mathcal{X} - Unpref(\mathcal{X}, M)\,] \cup Default(\mathcal{PR}, M)\,)$

where $\mathcal{PR} = \bigcup_{i \leq s}(P_i \cup R_i)$ *and* $\mathcal{X} = \mathcal{PR} - Reject(s, M, \Gamma)$.

Notice that the definition above requires M to be a stable model of Γ in order for M to be a preferred stable model of Γ. This reflects the requirement that preference information is intended to select among alternative stable models of Γ. In general, there may exist several preferred stable models. If the priority relation induced by the priority rules is a total order over the generalized rules in Γ, then Γ will admit at most one preferred stable model.

Example 2. (Contextual preferences) This example illustrates the use of contextual preferences to select preferred models. For simplicity, we employ only one state. Consider a scenario where John wants to buy a magazine. He can buy either a sport magazine (*sm*), a travel magazine (*tm*) or a financial magazine (*fm*). Suppose that John's preferences are context sensitive, i.e., he may have distinct preferences depending on time, location, etc. This example can be formalized via $\Gamma = \bigoplus\{(P_1, R_1)\}$, where:

$$P_1 = \left\{ \begin{array}{ll} sm \leftarrow not\, fm, not\, tm & (r_1) \\ tm \leftarrow not\, fm, not\, sm & (r_2) \\ fm \leftarrow not\, sm, not\, tm & (r_3) \end{array} \right\} \qquad R_1 = \left\{ \begin{array}{l} n_1 < n_3 \leftarrow holiday \\ n_2 < n_3 \leftarrow holiday \\ n_3 < n_1 \leftarrow office \\ n_3 < n_2 \leftarrow office \end{array} \right\}.$$

Since the priorities in R_1 cannot be used (both *holiday* and *office* being false), the preferred stable models of Γ coincide with its stable models (i.e., Γ has three models containing *sm*, *tm* and *fm*, respectively). If *holiday* would hold (for example via a subsequent update), then Γ would have two preferred stable models containing *sm* and *tm*, respectively.

5 Preferred Conclusions

The work briefly reviewed in the previous sections, proposes an approach for defining and reason with theories that capture preference information. Moreover, these theories are intended to model a dynamic world. Hence, the notion of update is also incorporated in the theory. In this section we address the problem of querying a dynamic prioritized program. A novel aspect is that queries can be equipped with the user's preferences and context information. That is, both context and preference information can be provided at query time. The ability to take into account the user information during the process of answering a query enhances the cooperative behaviour of the system. In fact, the system is now able to target its answers to the user's goal and interests.

Definition 13 (Query with preferences). *Let $s \in S$ be a state, G a (ground) goal, Σ a prioritized logic program, and $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ a dynamic prioritized program. Then $Q = (G, \Sigma)$ is a query w.r.t Γ.*

The aim of the prioritized logic program $\Sigma = (P, R)$ in a query $Q = (G, \Sigma)$ is to allow the user's knowledge or context information P to be taken into account when answering the query and also to allow the system to consider the user's preferences R[2]. Note that user knowledge evolution[3] could be easily captured by allowing Σ to be a dynamic prioritized program (instead of a prioritized logic program) in queries. Our framework could be easily extended to deal with this case. In the remainder, we assume that $S^+ = S \cup \{\max(S) + 1\}$.

Definition 14 (Joinability at state s). *Let $s \in S^+$ be a state. Given a dynamic prioritized program $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ and a prioritized logic program $\Sigma = (P_q, R_q)$, the joinability function at state s, written as $\diamond_s$, is defined as*

$$\Gamma \diamond_s \Sigma = \bigoplus\{(P'_i, R'_i) : i \in S^+\}$$

where

$$(P'_i, R'_i) = \begin{cases} (P_i, R_i) & \mathit{if}\ 1 \leq i < s, \\ (P_Q, R_Q) & \mathit{if}\ i = s, \\ (P_{i-1}, R_{i-1}) & \mathit{if}\ s < i \leq \mathit{max}(S^+). \end{cases}$$

From the definition above, the reader can see that joinability is parameterized by a state s. Its usefulness can be illustrated by the following examples. If we want to give priority to the user preferences in Σ over the preferences in Γ, then the joinability function $\diamond_{\max(S^+)}$ must be used. Several applications require this type of joinability, e.g., a web-site application of a travel agency whose database (formalized as a dynamic prioritized program) maintains information about holiday resorts and preferences among touristy locations, kinds of holidays, etc. When a user asks a query Q, the system has to give priority to the user interests and preferences expressed in Q.

On the other hand, other applications require, for instance, the joinability function $\diamond_1$. As an example, consider a program automating the procedure of allocating holiday periods to employees of a company. The database stores

[2] Flexible preferences in FQAS pose the problem of guaranteeing the respect of dynamic preferences for partial order postulates (antisymmetry, anti-reflexivity, transitivity). For reasons of space, this problem is not discussed here.

[3] For example, this ability allows us to model user evolving profiles (see [10] for a Web application).

information about holiday periods already allocated and preferences among alternative possibilities over periods of the year, e.g., it is preferable to have at least three weeks of holidays in August; in any other month of the year, it is preferable that not too many employees of the same division take holiday at the same time, etc. When a user asks a query, he provides preference information about the most suitable holiday periods for him or her, and the system must take them into account, but now the company preferences (expressed in the program) have more priority than the employee preferences.

Definition 15 (Preferred conclusions). *Let $s \in S^+$ be a state and $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ be a dynamic prioritized program. The preferred skeptical conclusions of Γ with joinability function $\diamond_s$ are*

$$\{Q \ : Q \text{ is a query } (G, \Sigma) \text{ and } G \text{ is included in every preferred stable model of } \Gamma \diamond_s \Sigma \text{ at state } max(S^+)\}\,.$$

The preferred credulous conclusions of Γ with joinability function $\diamond_s$ are

$$\{Q \ : Q \text{ is a query } (G, \Sigma) \text{ and } G \text{ is included in a preferred stable model of } \Gamma \diamond_s \Sigma \text{ at state } max(S^+)\}\,.$$

Example 3. Let Γ be the dynamic prioritized program $\bigoplus\{(P_1, R_1)\}$ of Example 2, and $\diamond_2$ a joinability function. Then
$Q_1 = (?{-}fm, (\{office\}, \{\}))$ is a skeptical preferred conclusion,
$Q_2 = (?{-}sm, (\{holiday\}, \{\}))$ is a credulous preferred conclusion, and
$Q_3 = (?{-}sm, (\{holiday, footballMatch\}, \{n_1 < n_2 \leftarrow footballMatch\}))$ is a skeptical preferred conclusion.

Definition 16 (Answer). *Let $s \in S^+$ be a state and $\Gamma = \bigoplus\{(P_i, R_i) : i \in S\}$ be a dynamic prioritized program. The skeptical (credulous) answer to the query $Q = (G, \Sigma)$ w.r.t Γ is* `yes` *if Q is a preferred skeptical (credulous) conclusion of Γ with joinability function $\diamond_s$. Otherwise, the answer is* `no`.

We have considered ground queries. But the notion of answer can be easily extended to non ground queries. As usual, an answer to a non ground query $Q = (G, \Sigma)$ (i.e. G is non ground) is the set of ground instances $\sigma(G)$ of G (σ is a substitution) such that the answer to $(\sigma(G), \Sigma)$ is `yes`.

Notice that every query $Q = (G, \Sigma)$ w.r.t a program Γ is evaluated in the most recent state of $\Gamma \diamond_s \Sigma$ (i.e., in the state $\max(S^+)$). However, in some applications it may be needed to evaluate queries in previous states (i.e., in the past). This capability could be introduced in our framework as well.

6 Example: Car Dealer

Consider the following (sketched) program that exemplifies the process of quoting prices for second-hand cars. For simplicity, the price of a car can be either 200 or 250 depending on how long the car has been in stock and on the colour (either orange or black) of the car. The policy of the company is to sell a car cheaper if the car has been in stock for a period equal or longer than 10 months, unless the customer likes the colour of the car. In such a case the company tries to sell the car with the (higher) price 250. Below we write non ground rules, and with that we intend their ground instances.

$$\begin{aligned} \mathit{price}(Car, 200) &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), \mathit{not}\; \mathit{price}(Car, 250), \mathit{not}\; \mathit{offer} && \text{(r1)}\\ \mathit{price}(Car, 250) &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), \mathit{not}\; \mathit{price}(Car, 200), \mathit{not}\; \mathit{offer} && \text{(r2)}\\ \mathit{prefer}(\mathit{orange}) &\leftarrow \mathit{not}\; \mathit{prefer}(\mathit{black}) && \text{(r3)}\\ \mathit{prefer}(\mathit{black}) &\leftarrow \mathit{not}\; \mathit{prefer}(\mathit{orange}) && \text{(r4)}\\ \mathit{stock}(Car, \mathit{Colour}, T) &\leftarrow \mathit{bought}(Car, \mathit{Colour}, \mathit{Date}), T = \mathit{today} - \mathit{Date} && \text{(r5)} \end{aligned}$$

When the company buys a car, the company must add (via an update) the car to the stock together with its colour and the date of purchase. For instance,

$$\mathit{bought}(\mathit{fiat}, \mathit{orange}, \mathit{d1}) \;\leftarrow$$

On the contrary, any time the company sells a car, the company must remove all the information about that car from the stock. For instance,

$$\mathit{not}\; \mathit{bought}(\mathit{volvo}, \mathit{black}, \mathit{d2}) \;\leftarrow$$

The following priority rules formalize the selling strategy of the company. By default, we assume that the customers prefer the colour black to orange.

$$\begin{aligned} n_2 < n_1 &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), T < 10\\ n_1 < n_2 &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), T \geq 10, \mathit{not}\; \mathit{prefer}(\mathit{Colour})\\ n_2 < n_1 &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), T \geq 10, \mathit{prefer}(\mathit{Colour})\\ \mathit{not}\; (n_3 < n_4) &\leftarrow\\ n_4 < n_3 &\leftarrow \end{aligned}$$

Suppose that later on the company adopts the policy to offer a special price for cars at certain times of the year. This is achieved via the update[4]

$$\begin{aligned} \mathit{price}(Car, 100) &\leftarrow \mathit{stock}(Car, \mathit{Colour}, T), \mathit{offer} && \text{(r6)}\\ \mathit{not}\; \mathit{offer} &\leftarrow \end{aligned}$$

As soon as *offer* holds (via a subsequent update), the rule (r6) will be in force by making the price of every car to be 100. Suppose that the company has in stock a fiat bought at date *d1* whose colour is orange, and that *offer* does not hold at the current state. Now, independently of the joinability function used, the answer to the query $Q1 = (?{-}\mathit{price}(\mathit{fiat}, P), (\{\}, \{\}))$ is $P = 250$ in case $\mathit{today} - \mathit{d1} < 10$, or $P = 200$ in case $\mathit{today} - \mathit{d1} \geq 10$. Things change if a customer asks the query $Q2 = (?{-}\mathit{price}(\mathit{fiat}, P), (\{\}, \{\mathit{not}\,(n_4 < n_3), n_3 < n_4\}))$, stating that he prefers the colour orange. The intended answer is $P = 250$ irrespectively of how

[4] Note that we can express more complex rules for pricing cars depending for example on whether or not the customer wants to sell his old car. In such a case, the customer must specify (in P_Q) at query time the kind of car he wants to sell.

long the car has been in stock. Note that for this query it is relevant which joinability function is used. In fact, if we use $\diamond_1$, then we do not get the intended answer because the user preferences (regarding the preferred colour) would be overwritten by the default preferences of the company. On the other hand, it would not be so appropriate (from the company's perspective) to use $\diamond_{\max(S^+)}$. Indeed, a customer could ask $Q3 = (?{-}\mathit{price}(\mathit{fiat}, P), (\{\mathit{offer}\}, \{\}))$.

7 Conclusions and Future Work

We have presented a novel logical framework where update and preference information can be specified and used in query answering systems. We have shown through some examples how the proposed approach can enhance the cooperative behaviour of the system. The procedural semantics of the approach is based on a syntactic transformation presented in [1].[5]

The approach can be extended in several ways. For example, the user knowledge evolution could be easily captured by allowing Σ (in queries) to be a dynamic prioritized program. Another extension would be to equip queries with a state to allow the users to query the past. An interesting line of research is to investigate (1) how to incorporate abduction into our framework: we may have abductive preferences leading to conditional answers depending on accepting a preference; and (2) how to tackle the problem arising when we have several users query the system together (i.e., when we need to take into account all their preferences). From a practical point of view, we may fornish the user with a preference metalanguage that compiles into the tedious pairwise preference specification. Additionally, we can detect inconsistent preference specifications and cater for interactive removal of such specifications.

References

1. J. J. Alferes, P. Dell'Acqua, and L. M. Pereira. A compilation of updates plus preferences. To appear in: *Logic in Artificial Intelligence* (Jelia'02), LNAI, 2002.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *The J. of Logic Programming*, 45(1-3):43–70, 2000. A short version titled *Dynamic Logic Programming* appeared in A. Cohn and L. Schubert (eds.), *KR'98*, Morgan Kaufmann.
3. J. J. Alferes and L. M. Pereira. Updates plus preferences. In M. O. Aciego et al. (eds.), *Logics in AI, Procs. JELIA'00*, LNAI 1919, pp. 345–360, Berlin, 2000.
4. G. Brewka. Well-founded semantics for extended logic programs with dynamic preferences. *Journal of Artificial Intelligence Research*, 4, 1996.
5. G. Brewka and T. Eiter. Preferred answer sets for extended logic programs. *Artificial Intelligence*, 109, 1999.
6. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. Bowen (eds.), *Proc. of the Fifth Int. Conf. on Logic Programming*, pp. 1070–1080, Cambridge, MA, 1988. The MIT Press.

[5] A prototype implementation of updates plus preferences implemented by J. J. Alferes is available at: http://centria.di.fct.unl.pt/~jja/updates.

7. K. Inoue and C. Sakama. Negation as failure in the head. *Journal of Logic Programming*, 35:39–78, 1998.
8. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In B. Nebel et al. (eds.), *KR'92*. Morgan-Kaufmann, 1992.
9. J. Minker. An overview of cooperative answering in databases. (Invited paper.) Int. Conf. on Flexible Query Answering Systems (FQAS), 1998.
10. J. Pagonis and M. C. Sinclair. Evolving user profiles to reduce internet information overload. In R. John and R. Birkenhead (eds.), *Int. Conf. on Recent Advances in Soft Computing*, Advances in Soft Computing 232, pp. 100–107. Springer, 2000.

Bipolarity in Flexible Querying

Didier Dubois and Henri Prade

Institut de Recherche en Informatique de Toulouse (I.R.I.T.) – C.N.R.S.
Université Paul Sabatier, 118 route de Narbonne
31062 Toulouse Cedex 4, France
{dubois, prade}@irit.fr

Abstract. The paper advocates the interest of distinguishing between negative and positive preferences in the processing of flexible queries. Negative preferences express what is (more or less, or completely) impossible or undesirable, and by complementation state flexible constraints restricting the possible or acceptable values. Positive preferences are not compulsory, but rather express wishes; they state what attribute values would be really satisfactory. The paper discusses the handling of bipolar queries, i.e. queries involving negative and positive preferences, in the framework of possibility theory. Both ordinary queries expressed in terms of requirements, and case-based queries referring to examples, are considered in this perspective.

1 Introduction

Flexible queries have aroused an increasing interest for many years in the database literature (Bosc and Pivert, 1992; Christiansen *et al*., 1997; Larsen *et al*. 2001), and the fuzzy set-based approach, which was introduced about 25 years ago, has been developed both at the theoretical and the practical levels through many works (e.g., Bosc and Kacprzyk, 1995; Petry, 1996).

In these works, flexible queries have been generally thought in terms of conjunctions of constraints restricting possible values of attributes. By flexible queries, we mean that these constraints could be fuzzy, or prioritized. Fuzzy constraints can be viewed as preference profiles: values associated with degree 1 are fully satisfactory, while values with degree 0 are completely rejected; the smaller the degree, the less acceptable the value. The violation, by a possible solution, of a (crisp) prioritized constraint leads to a non-zero upper bound of the global evaluation of the set of constraints, when the priority of this constraint is not maximal; the upper bound (say $1 - \alpha$) is all the smaller as the constraint has a higher priority (α). Fuzzy constraints can be viewed as collections of nested prioritized constraints. Fuzzy constraints correspond to "negative" preferences in the sense that their complements define fuzzy sets of values that are rejected as being non-acceptable. These constraints should be combined conjunctively, thus acknowledging the fact they are constraints. However, there is another type of preferences, which will be qualified as 'positive' in the following. These preferences are not constraints, but only wishes, which are more or less strong. If at least some of these wishes are satisfied, it should give some bonus to the corresponding solutions (provided that they also satisfy the constraints, if any). Wishes are not compulsory and can be combined disjunctively.

Recently, Benferhat *et al.* (2002) have proposed a bipolar possibilistic logic framework for modelling preferences. On the one hand, prioritized logical formulas (weighted in terms of necessity degrees) are used for expressing constraints whose priority are more or less high, which thus delimit the fuzzy set of possible worlds compatible with the constraints. On the other hand, other formulas, weighted in terms of a "guaranteed possibility" function, express the minimal level of satisfaction reached if the value under concern lies in some subsets of interpretations. The second type of formulas corresponds to a "positive" assessment of what is wished, while the first type expresses what is allowed, and rather reflects the result of "negative" preferences (what is rejected defines, by complementation, what may be acceptable). The consistency of the two types of preferences require that the fuzzy set of the interpretations which have some guaranteed satisfaction level is included in the fuzzy set of the interpretations which are compatible with the constraints.

This paper offers a tentative discussion of the idea of bipolarity in flexible querying. Two lines of research are considered. The first one, in Section 3, considers queries (e. g., looking for an apartment) involving both fuzzy constraints (e. g., "not too expensive", this is a constraint, since the user cannot afford to pay a too high fare), and fuzzy wishes (e.g., preferably "near the train station"). Section 4 studies another situation where bipolarity takes place, namely in case-based querying, where the query is evaluated both on the basis of the similarity to examples of what is looked for, and on the basis of the dissimilarity w. r. t. counter-examples. The next section first discusses the basic notions for the bipolar representation of preferences.

2 Bipolar Representation of Preferences

It has been observed for a long time that humans express both negative and positive judgements. This may apply to beliefs as well as preferences. For instance, Buchanan and Shortliffe (1984) found it natural to assess the uncertainty of an hypothetical statement, given the available evidence, by means of a pair made of a 'measure of belief' and a 'measure of disbelief' (then combined into a certainty factor). For preferences, one states what is desired on the one hand, as well as what is rejected on the other hand. But what is not rejected is not necessarily desired, nor what is not desired necessarily rejected.

Let U be a set of possible choices or solutions (it may be a set of objects, or the domain of an attribute used for describing objects). If we assume for simplicity, for the moment, that preferences are $\{0, 1\}$-valued, it means that U in general is partitioned in three subsets, D, the set of desirable choices, R, the set of rejected choices, and $U - (D \cup R)$. Consistency of preferences entails that $D \cap R = \varnothing$. Moreover, since one may be indifferent to some choices, one often has $U - (D \cup R) \neq \varnothing$.

From D and R, we can equivalently work with the nested pair $(D, U - R)$ where $D \subseteq U - R$. When every possible choice is either desired or rejected, we have $D = U - R$. However, the set of desired choices D is usually only a proper subset of the set $U - R$ of non-rejected choices.

This simple representation framework can be straightforwardly extended to graded preferences. We shall use [0, 1] as the preference scale. D and R are then fuzzy sets such as

$$\sup_u \min(\mu_D(u), \mu_R(u)) = 0 \text{ (consistency)} \qquad (1)$$

which ensures

$$\forall u, \mu_D(u) \leq 1 - \mu_R(u) \qquad (2)$$

The pair (R, D) under condition (2) is an "intuitionistic fuzzy set" in the sense of Atanassov (1986), while the pair $(D, U - R)$ under condition (1) is a twofold fuzzy set (Dubois and Prade, 1987) since support $(D) = \{u, \mu_D(u) > 0\} \subseteq \text{core}(R^c) = \{u, \mu_R(u) = 0\}$ where R^c is the fuzzy complement of R.

Let us suppose now that, depending on criteria, or on sources i, we are faced with n pairs (D_i, R_i^c), to be fused into a pair (D, R^c). The D_i's have to combine disjunctively since if u is desirable according to i, it should be desirable overall. This is the same with the R_i's: if u is rejected according to i, it should be rejected overall. Thus,

$$(D, R^c) = (\bigcup_i D_i, \bigcap_i (R_i^c)) \qquad \text{with } R = \bigcup_i R_i \qquad (3)$$

where $\bigcap_i$ and $\bigcup_i$ stands for fuzzy set intersection and union. They can be respectively defined by min and max operations. Other triangular norms and co-norms could be used if a reinforcement effect is needed (e.g., if a value u has a rather small acceptability degree with respect to two constraints $C_i = R_i^c$ and $C_j = R_j^c$ then its acceptability degree w.r.t. the conjunction of the constraints can still be much smaller, or even zero).

3 Bipolar Queries

Bipolar queries are queries, which involve two components, one pertaining to simple wishes, and the other to constraints. Let i = 1, n be a set of attributes. Let C_i be the subset of the domain U_i of attribute i, representing the (flexible) constraint restricting the acceptable values for i (in other words, $C_i = R_i^c$ where R_i is the (fuzzy) set of rejected values according to i). Let D_i be the subset of the domain of i, representing the values, which are really wished and satisfactory for i. The consistency condition (2) is assumed for each pair (C_i, D_i), i.e.,

$$\forall u, \mu_{D_i}(u) \leq \mu_{C_i}(u) \quad (4)$$

expressing that a value cannot be more wished than it is allowed by the constraint. Then a query is represented by a set of pairs $\{(C_i, D_i), i = 1, n\}$ satisfying (4).

It may happen that for some i there is no constraint. In such a case $C_i = U_i$, i.e.,$\forall u$, $\mu_{C_i}(u) = 1$. If no value is particularly wished in U_i then $D_i = \varnothing$, i.e.,$\forall u$, $\mu_{D_i}(u) = 0$. For example, the query "find an apartment not too expensive, preferably near the train station" involves two attributes, the price, and the distance to the station. Let us assume that one wants to express a constraint on the price and a wish on the distance to the station. This is represented by the set of the two pairs {(Not_too_expensive, $\varnothing$), ($U_{distance}$, Near)}, where Not_too_expensive and Near are labels of fuzzy sets. This means that any item in the database will not be rejected inasmuch it satisfies the price constraint, and that it will be really satisfactory if moreover it is near the station.

Then following (3), the natural aggregation scheme for a query represented by a set of pairs $\{(C_i, D_i), i = 1, n\}$, with possibly $C_i = U_i$ or $D_i = \varnothing$ is given by

$$(C, D) = (\times_i C_i, +_i D_i) \qquad (5)$$

where $\times_i$ denotes the Cartesian product on $U_1 \times \ldots \times U_n$ and $+_i$ the Cartesian co-product (i.e., $D_i + D_j = ((D_i)^c \times (D_j)^c)^c$). Note that we have $+_i D_i \not\subset \times_i C_i$ generally, even if (4) holds for each i. The consistency of the wishes with the constraints can be formally recovered by stating

$$(C, D) = ((\times_i C_i), (+_i D_i) \cap (\times_i C_i)). \qquad (6)$$

How can a query $\{(C_i, D_i), i = 1, n\}$ be evaluated? Given a tuple $u = (u_1, \ldots, u_i, \ldots, u_n)$ in an ordinary database, we can thus compute a pair of degrees of match, namely

$$(C(u), D(u)) = (\min_i \mu_{C_i}(u_i), \max_i \mu_{D_i}(u_i)), \qquad (7)$$

which reflects the extent to which u satisfies all the constraints and at least one wish.

The question is then to rank-order the tuples. A basic idea is to give priority to constraints, and thus to have a lexicographic ranking, by using C(u) as primary criterion, and D(u) as a secondary one for breaking ties. This procedure can be improved by considering that satisfying several wishes is certainly better than satisfying just one. Let σ be a permutation reordering the $\mu_{D_i}(u_i)$'s decreasingly, i.e.

$$\mu_{D_{\sigma(1)}}(u_{\sigma(1)}) \geq \ldots \geq \mu_{D_{\sigma(n)}}(u_{\sigma(n)}).$$

Note that if no wish is expressed, $D_i = \varnothing$, then $\mu_{D_i}(u_i) = 0\ \forall u_i$, and the evaluation is ranked at the end. This leads to rank–order the vectors

$$(\min_i \mu_{C_i}(u_i), \mu_{D_{\sigma(1)}}(u_{\sigma(1)}), \ldots , \mu_{D_{\sigma(n)}}(u_{\sigma(n)})) \qquad (8)$$

lexicographically from (1, 1, … , 1) to (0, 0, … , 0). This method is faithful w.r.t. the fuzzy cardinality of the fuzzy set of wishes which are somewhat satisfied (see Dubois, Fargier and Prade, 1996). Another way, which allows for compensation between the levels of satisfaction of the different wishes, would be to compute the fuzzy set cardinality $\sum_i \mu_{D_i}(u_i)$ and to use it as a secondary ranking criterion for breaking ties w.r.t. C(u), as above. Clearly, some information is then lost by the scalar cardinality which makes no difference between two distinct situations: i) few highly satisfied wishes, or ii) many wishes satisfied at lower degrees. However, using (8), the lexicographical method will rank-order cases corresponding to the situation (i) first, and may even rank a tuple u before another one u', which are such that C(u) = C(u'), while we have $\sum_i \mu_{D_i}(u_i) < \sum_i \mu_{D_i}(u'_i)$ (e.g., $\mu_{D_1}(u_1) = 1$ and $\mu_{D_i}(u_i) = 0$ for i=2,n, while $\mu_{D_i}(u'_i) = 0.5$ for i=1, n and $n \geq 3$).

Whatever the secondary ranking criterion that is used, a query, based on the distinction between negative and positive desires, cannot be made equivalent to a unipolar query where positive desires would be also expressed by means of (possibly weakened) constraints. For instance, it is not the same to look for "an apartment not too expensive, preferably near the train station", or for "an apartment not too expensive, and not too far from the train station". In the second query, any apartment that is not sufficiently close to the station is definitively rejected (and thus not retrieved); this is not the case with the first query.

Remark
We might think of using a unique scalar ranking index, although based on a bipolar view. As we are going to see in this remark, two ways that we may think of for doing it, turn to be unsatisfactory.

A first idea would be, inside the fuzzy set of tuples satisfying the constraints C_i, to favor those satisfying some D_i, or at least to discount those which satisfy the D_i's too poorly. The following expression offers such behavior

$$E(u) = \min(C(u), \lambda \cdot C(u) + (1 - \lambda) \cdot D(u)), \ (9)$$

with $C(u) = \min_i \mu_{C_i}(u_i)$ and $D(u) = \perp_i \mu_{D_i}(u_i)$ where $\perp$ is a disjunctive aggregation operator. As it can be seen, (9) offers the possibility of some trade-off controlled by the parameter $\lambda > 0$, while $E(u) = C(u)$ as soon as $D(u) > C(u)$. E(u) is decreased below C(u) if the value of D(u) is too small. Note that however $E(u) \neq 0 \Leftrightarrow C(u) \neq 0$. Moreover the same evaluation $E(u) = \lambda$ is obtained for $(C(u), D(u)) = (\lambda, \lambda)$ and for $(C(u), D(u)) = (1, 0)$. This suggests to have λ rather high. For instance, if we consider the two vectors $(C(u), D(u)) = (.8, 1)$ and $(C(u'), D(u')) = (.9, 0)$, we have for $\lambda = .7$, $E(u) = \min(.8, .7 \cdot .8 + .3 \cdot 1) = .8$, while $E(u') = \min(.9, .7 \cdot .9 + .3 \cdot 0) = .63$. This example illustrates the effect of (9) which ranks u before u' (while the lexicographic method rank-orders u' first). This may be found satisfying, but having the same evaluation for $(C(u''), D(u'')) = (.63, 1)$ and $(C(u'), D(u')) = (.9, 0)$ may be found more debatable.

Another idea would be to have a hierarchical aggregation between C and D. Namely C should be satisfied first, and among the tuples satisfying C (if any) the ones satisfying D are preferred. But satisfying D if C is not satisfied is of no interest. Thus, one would like to express that C should hold (with priority 1), and that if C holds, D holds with some priority α. This leads to the following evaluation

$$E(u) = \min(C(u)\ \max(D(u), 1 - \min(C(u), \alpha))).$$

In the above expression, it is easy to check that:

$$C(u) \leq 1 - \alpha \Rightarrow E'(u) = C(u)$$

$$C(u) \geq 1 - \alpha \text{ and } D(u) \leq 1 - \alpha \Rightarrow E'(u) = 1 - \alpha$$

$$C(u) \geq 1 - \alpha \text{ and } D(u) \geq 1 - \alpha \Rightarrow E(u) = \min(C(u), D(u)).$$

Thus E'(u) behaves as an ordinary conjunction for high values of C(u) and D(u)), while $E'(u) = C(u)$ when C(u) is below some threshold $1 - \alpha$. However, when C(u) is high, and D(u) below the threshold, E'(u) remains constant at a value somewhat smaller than C(u), which is not really satisfactory (since E'(u) cannot increase with C(u) in this area).

4 Bipolarity in Case-Based Querying

The user is not always able to easily express a request by expressing restrictions on attribute domains, even in a flexible way, these restrictions being then conjunctively combined. It may be more convenient for him to express what he is looking for from prototypical examples, still keeping the idea of positive and negative requirements.

Suppose that, as in the system described in (de Calmès *et al.*, 2002), the user is looking for some houses to let, described in a database. One may think of providing the user with typical examples of existing houses stored in the base, and which are representative of different categories. The system may also propose to the user a small set of houses (may be fictitious, but realistic), which are well-contrasted on the attributes of interest for the user. We may also, more simply, assume that the examples are generated by the user himself. They may be examples of what he likes, or examples of what he dislikes. The user may also refer to existing cases he already experienced if any (i.e. houses he rent in the past, in our example).

The user is supposed to say to what extent a few examples of items are representative of what he is looking for by using some (finite) preference scale. Querying based on examples, for eliciting the user's preferences, can thus provide the necessary information for building a query. Then, the relevance of stored items should be evaluated in terms of their similarity with respect to these preferred examples (and their dissimilarity from counter-examples he dislikes). These issues are clearly close to fuzzy case-based reasoning.

Examples, as well as counter-examples, are supposed to be described in terms of precisely known attribute values. These attributes are assumed to be relevant and sufficient for describing their main features from the user's point of view. Then a request should look for the items which are similar to at least one example (w.r.t. all the attributes) and which are dissimilar to all the counter-examples (each time w.r.t. at least one attribute), as proposed in (Dubois, Prade, and Sèdes, 2001).

Moreover we can take into account the extent to which each example or counter-example is highly representative, and the importance of the (relevant) attributes. Thus, the more *similar* a tuple *u* is w.r.t. *(at least)* a representative example and the more *dissimilar u* is w.r.t. *all* representative counter-examples, the more *possible u* is eligible for the user. This evaluation might be also improved by requiring the similarity of u wit *several* examples when possible (see (Dubois *et al.*, 1988) for the handling of softly quantified weighted min expressions).

Suppose for simplicity that all the importance and representativeness weights are equal; see (Dubois, Prade, and Sèdes, 2001) for the general case. The items u are then rank-ordered in terms of the function

$$E''(u) = \min[\max_j \min_i S_i(a_i(u), a_{ij}), \min_k \max_i (1 - S_i(a_i(u), b_{ik}))] \qquad (10)$$

where $a_i(u)$ is the value of attribute *i* for item *u*, a_{ij} is the value of attribute *i* for example *j*, b_{ik} the value of attribute *i* for counter-example *k*, where S_i is a fuzzy similarity relation on the attribute domain of *i* (S_i is supposed to be reflexive and symmetric). Clearly, $S_i(\cdot, a_{ij})$ defines a fuzzy set of values close to a_{ij}, while $1\text{-}S_i(\cdot, b_{ik})$ defines a fuzzy set of values not similar to b_{ik}, for each attribute *i* (it may be replaced by a smaller fuzzy set of values *significantly* different from b_{ik}).

As it can be seen in (10), the positive examples are combined disjunctively as expected, since the user may like houses which are quite different. The negative examples which are disliked, in agreement with (3), are also combined disjunctively, but after complementation appears in a conjunctive combination in (10).

Note also that the above expression may incorporate *interactivity* between attributes. For instance, the user may be simultaneously interested in two types of houses, one which is 'small' with a 'low' price, and another which is 'large' but with a 'medium' price. Such a type of requirement is more easily expressed by referring to examples, than by expressing compound requirements on attribute values.

5 Concluding Remarks

This paper has primarily emphasized the interest of distinguishing between rejected values and desired values in the expression of preferences, as acknowledged by many studies in cognitive psychology. This is clearly important when dealing with flexible queries. Rejected values lead by complementation to constraints, while positive wishes should be independently considered for refining choices between data satisfying the constraints.

The paper has illustrated this bipolar view of queries both on attribute-based querying and on example-based querying in face of an ordinary database. The handling of ill known data in this perspective is a topic for further research. Besides, we may think of hybrid queries made of examples and of classical (fuzzy) restrictions expressing what attribute values are undesirable.

It is worth noticing that in this approach the evaluations pertaining to positive and negative preferences are not recombined in a unique evaluation, as in some approaches for the handling of bipolar criteria (Grabisch and Labreuche, 2000). A separate treatment of positive and negative preferences in decision making has been also recently proposed by Fortemps and Slowinski (2002), but in a relational framework based on the pairwise comparison of alternatives.

References

1. K. T. Atanassov (1986) Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, **20**, 87-96,1986.
2. S. Benferhat, D. Dubois, S. Kaci, H. Prade (2002)Bipolar representation and fusion of preferences in the possibilistic logic framework. *Proc. of the 8th International Conference on Principles of Knowledge Representation and Reasoning,* April 22-25, Toulouse, France. Morgan Kaufmann Publi., 421-432.
3. P. Bosc, J. Kacprzyk (Eds.) (1995) *Fuzziness in Database Management Systems*. Physica-Verlag, Heidelberg.
4. P. Bosc, O. Pivert (1992) Some approaches for relational databases flexible querying. *J. of Intelligent Information Systems*, 1, 323-354.

5. B. G. Buchanan, E. H. Shortliffe (1984) *Rule-Based Expert Systems – The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA.
6. M. de Calmès, D. Dubois, E. Hüllermeier, H. Prade, F. Sèdes (2002) A fuzzy set approach to flexible case-based querying: methodology and experimentation. *Proc. of the 8th International Conference, Principles of Knowledge Representation and Reasoning* (KR2002) , Toulouse, France, April 22-25, 2002, Morgan Kaufmann Publishers , San Francisco, Ca , 449-458.
7. H. Christiansen, H.L. Larsen, T. Andreasen (Eds.) (1997) *Flexible Query Answering Systems*, Kluwer Academic Publishers.
8. D. Dubois, H. Fargier, H. Prade (1996) Refinements of the maximin approach to decision-making in fuzzy environment. *Fuzzy Sets and Systems*, **81**, 103-122.
9. D. Dubois, H. Prade Twofold fuzzy sets and rough sets - Some issues in knowledge representation. *Fuzzy Sets and Systems*, **23**, 3-18,1987.
10. D. Dubois, H. Prade, C. Testemale (1988) Weighted fuzzy pattern matching, *Fuzzy Sets and Systems*, **28**, 313-331.
11. P. Fortemps, R. Slowinski (2002) A graded quadrivalent logic for ordinal preference modeling: Loyola-like approach. *Fuzzy Optimization and Decision Making*, **1**, 93-111.
12. M. Grabisch, C. Labreuche (2000) The Sipos integral for the aggregation of interacting bipolar criteria. *Proc. 8th Inter. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems* (IPMU'00), Madrid, 395-409.
13. H. Larsen, J. Kacpryk, S. Zadrozny, T. Andreasen, H. Christiansen (Eds.) (2001) *Flexible Query Answering Systems, Recent Advances*, Physica Verlag.
14. F.E. Petry (1996) *Fuzzy Databases: Principles and Applications.* Kluwer Acad. Pub.

Theory of K-Calculuses as a Powerful and Flexible Mathematical Framework for Building Ontologies and Designing Natural Language Processing Systems

Vladimir A. Fomichov[1,2]

[1]Faculty of Applied Mathematics,
Moscow State Institute of Electronics and Mathematics (Technical University),
109028 Moscow, Russia
[2]Department of Information Technologies
K.E.Tsiolkovsky Russian State Technological University - "MATI",
121552 Moscow, Russia
vdrfom@aha.ru

Abstract. The necessity of developing more powerful and convenient formal means (in comparison with the widely used ones) for describing structured meanings of natural language (NL) texts and building ontologies is grounded. The basic principles of an original mathematical approach to this problem are outlined; this approach is given by the the theory of K-calculuses and K-languages (the KCL-theory) elaborated by the author. The considered examples of building semantic representations of the NL-texts, of constructing definitions of concepts, and of finding an answer to a question of the end user pertain to biology, medicine, ecology, and business. The advantages of the KCL-theory in comparison with Discourse Representation Theory, Theory of Conceptual Graphs, and Episodic Logic are stated. It is reported about a number of successful applications of the KCL-theory to the design of NL-processing systems. The metaphor "a kitchen combine" is used for underlining that the KCL-theory provides a broad spectrum of new opportunities for the design of NL-processing systems and building ontologies.

1 Introduction

The dominant part of knowledge obtained by the mankind is stored in the form of texts in natural language (NL). The progress in hardware of computers has provided technical preconditions for developing textual (or full-text) databases (TDBs). However, adequate logical foundations of building TDBs are still to be elaborated. At least the following tasks raise new demands to Mathematical Computer Science, Mathematical Linguistics, and Cybernetics: text summarization; conceptual information retrieval in TDBs on requests of end users; extracting information from textual sources for forming and up-dating knowledge bases of applied intelligent systems, and the creation of Semantic Web. These tasks require such mathematical means which would be convenient for studying the regularities of conceptual processing of arbitrary NL-texts pertaining to science, technology, law, medicine, business, etc. During recent years, the role of ontologies in understanding NL-texts and supporting the dialogues in NL has been widely recognized. With respect to the role of ontologies in NL-processing, the analysis carried out in [1-3] indicated that we need, first of all, the mathematical means being convenient both for describing conceptual structures (or semantic structures, or structured meanings) of arbitrary real NL-texts and for representing knowledge about the world. Let's call the problem of creating such means as the problem PR1.

In the field of designing NL-processing systems (NLPSs), the most popular approaches to formal studying semantics of NL are Discourse Representation Theory, or DRT [4, 5], Theory of Conceptual Graphs, or TCG [6, 7], Episodic Logic, or EL [8-10]. Each of these approaches contributed to formal studying some aspects of NL semantics. However, all enumerated approaches, except of EL, don't consider the problem PR1 and, as a result, are very far from solving it. Though EL is not a strictly mathematical approach, it suggests a number of precious ideas for describing conceptual structures of discourses and representing knowledge about the world. Nevertheless, EL is far from solving the problem PR1 too, in particular, because it doesn't provide effective formal means for: building formal analogues of concepts and representing operations on concepts as, e.g., in KL-ONE-like languages; describing conceptual microstructure of discourses (a) with references to the meanings of phrases and larger parts of texts, (b) with complicated designations of sets, (c) using logical connectives for joining designations of concepts or infinitives with dependent words or designations of things or designations of sets.

Let's consider, as an example of an important practical task, the problem of computational processing of business documents. The texts from such documents may include:

(a) questions with interrogative words; (b) questions with the answer "Yes" or No";
c) infinitives with dependent words ("to sell 50 boxes with oranges");
(d) constructions formed out of infinitives with dependent words by means of the logical connectives "and", "or", "not";
(e) complicated designations of sets ("a consignment consisting of 50 boxes with oranges");
(f) fragments where the logical connectives "and", "or" join not the designations of assertions but the designations of objects ("the product A is distributed by the firms B1, B2, …, BN");
(g) explanations of the terms being unknown to an applied intelligent system (because the firms invent and produce new products);
(h) fragments containing the references to the meanings of phrases or larger fragments of a discourse ("this proposal", "that order", etc.);
(i) the designations of the functions whose arguments and/or values may be the sets of objects ("the staff of the firm A", "the suppliers of the firm A", "the number of the suppliers of the firm A").

The analysis shows that the expressive power of DRT, TCG, and EL is insufficient for effective representing contents of arbitrary business documents. In particular, from the standpoint of describing semantic structure of the fragments of the types (d) - (h) and taking into account the existence of the designations of the type (i). That is why there is the necessity of elaborating much more powerful and flexible formal means allowing for representing contents of arbitrary business documents. This applies also to formal representing contents of arbitrary texts in medicine, law, economy, etc.

Meanwhile, it appears that the first solution of the problem PR1 does exist already and is provided by a new mathematical approach to studying semantics and pragmatics of NL. It was developed in Moscow and is called Integral Formal Semantics (IFS) of NL. The basic philosophical principles of IFS are published in English, in particular, in [1]. The central constituent of IFS is the theory of K-calculuses (knowledge calculuses) and K-languages (the KCL-theory) represented, in particular, in [1-3, 11, 12]. The KCL-theory is an original theory of describing in a mathematical way the structured meanings (SMs) of sentences and complicated discourses in NL, representing knowledge about the world and goals of intelligent systems, and of describing the correspondence between NL-texts and their SMs. An important part of the KCL-theory is the theory of restricted K-calculuses and K-languages (the RKCL-theory) set forth, in particular, in [3, 12].

The formal purpose of this paper is as follows: (a) to give an outline of the RKCL-theory, (b) to demonstrate important new opportunities provided by the RKCL-theory for building ontologies and describing SMs of real NL-texts pertaining to science and business, (c) to show the possibilities of using the formulas of K-calculuses for representing linguistic metaknowledge enabling to find a referent of a word combination in such cases when this referent in not explicitly indicated in the phrase containing the word combination, (d) to illustrate the combined use of knowledge about the world and linguistic metaknowledge in the process of finding an answer to a question of the end user.

The informal purpose of the paper consists in attracting the attention of the designers of NL-processing systems to the facts that (a) there exist now much more powerful formal means for constructing TDBs and for building ontologies than the means widely used with this aim, (b) these new means are able to work effectively in studying arbitrary NL-texts and elaborating ontologies in arbitrary application domains.

2 A Kitchen Combine as a Metaphor Concerning the Theory of K-Calculuses and K-Languages

It seems that a metaphor is able to better grasp the significance of the KCL-theory for the designers of NLPSs. It establishes a connection between the problems of a housekeeping and the problems of designing a NLPS.

When a woman having a full-time job enters the kitchen, she has a lot of things to do in a short time. The kitchen combines are constructed in order to make the work in the kitchen easier and to diminish the time needed for the work of the kind. For this, the kitchen combines can chop, slice, stir, grate, blend, squeeze, grind, beat.

The designers of NLPSs have a lot of things to do in a very restricted time. That is why they need effective formal tools for this work. Like a kitchen combine for housekeeping, the KCL-theory can help the designers of NLPSs to do many things. In particular, the KCl-theory is convenient for:

- constructing formal definitions of concepts,
- representing knowledge associated with concepts,

- building knowledge modules (in particular, definitions of concepts) as the units having both the content (e.g., a definition of a concept) and the external characteristics (e.g., the authors, the date of publishing, the application fields),
- representing the goals of intelligent systems,
- building semantic representations of various algorithms given in a natural language form,
- representing the intermediate results of semantic-syntactic processing of NL-texts (in other words, building underspecified semantic representations of the texts),
- forming final semantic representations of NL- texts,
- representing the conceptual macro-structure of a NL discourse,
- representing the speech acts,
- building high-level conceptual descriptions of the figures occurred in the scientific papers, textbooks, technical patents, etc.

No other theory in the field of formal semantics of NL can be considered as a useful tool for all enumerated tasks. In particular, it applies to Montague Grammar and its extensions, DRT, TCG, EL.

In case of technical systems, a highly precious feature is the simplicity of construction. Very often, this feature contributes to the reliability of the system and the easiness of its exploitation. The KCL-theory satisfies this criterion too, because it makes the following discovery both in non-mathematical and mathematical linguistics: a system of such 10 operations on structured meanings (SMs) of NL-texts is found that, using primary conceptual items as "blocks", we are able to build SMs of arbitrary NL-texts (including articles, textbooks, etc.) and arbitrary pieces of knowledge about the world. Such operations will be called *quasilinguistic conceptual operations.* Hence the KCL-theory suggests a *complete collection* of quasilinguistic conceptual operations (it is a hypothesis supported by many weighty arguments).

A complete system of mathematical definitions determining such collection of conceptual operations can be found in Fomichov [3]. This system of definitions determines also a class of formal languages called restricted standard K-languages (RSK-languages). The useful properties of the KCL-theory stated above allow for the conclusion that the KCL-theory can be at least not less useful for the designers of NLPSs as a kitchen combine is of use for making easier the work in the kitchen.

3 Briefly about the Inference Rules of Restricted K-Calculuses

Let's consider the main ideas of determining a new class of formal systems called restricted K-calculuses and, as a consequence, a new class of formal languages called RSK-languages. The exact mathematical definitions can be found in [3].

At the first step (consisting of a rather long sequence of auxiliary steps), the RKCL-theory defines a class of formal objects called *simplified conceptual bases (s.c.b.).* Each s.c.b. *B* is a system of the form $((c_1, c_2, c_3), (c_4,..., c_7), (c_8,..., c_{14}))$ with the components $c_1,..., c_{14}$ being mainly finite or countable sets of symbols and distinguished elements of such sets. In particular, $c_1 = St$ is a finite set of symbols called sorts and designating the most general considered concepts; $c_2 = P$ is a distinguished sort "sense of proposition"; $c_4 = X$ is a countable set of strings used as elementary blocks for building knowledge modules and semantic representations (SRs) of texts; X is called a primary informational universe; $c_5 = V$ is a countable set of variables; $c_7 = F$ is a subset of X whose elements are called functional symbols.

Each s.c.b. *B* determines three classes of formulas, where the first class *Lrs(B)* is considered as the principal one and is called *the restricted standard K-language in the s.c.b. B*. Its strings (they are called K-strings, or l-formulas) are convenient for building semantic representations (SRs) of NL-texts. We'll consider below only the formulas from the first class *Lrs(B).*

In order to determine for arbitrary s.c.b. *B* three classes of formulas, a collection of inference rules P[0], P[1],..., P[10] is defined. The ordered pair *Krs(B) = (B, Rls),* where *Rls* is the set consisting of all these rules, is called *the restricted K-calculus in the s.c.b. B*. The rule P[0] provides an initial stock of formulas from the first and second classes. E.g., there is such a s.c.b. B_1 that, according to the rule P[0], $Lrs(B_1)$ includes the elements *box1, green, city, set, India, 7, all, any, Weight, Distance, Staff, Suppliers, Quantity, x1, x2, P1, P2, Manufactured-in, delivery, Addressee, "Spencer & Co.", Propose, Want.*

Let's regard (ignoring many details) the structure of strings which can be obtained by applying any of the rules P[1],..., P[10] at the last step of inferencing these formulas. The rule P[1] enables us to build l-formulas of the form *Quant Conc* where *Quant* is a semantic item corresponding to the meanings of such words and expressions as "certain", "any", "arbitrary", "each", "all", "several", "many", etc. (such semantic items will be called *intensional quantifiers*), and *Conc* is a designation (simple or compound) of a concept. The examples of K-strings for P[1] as the last applied rule are as follows:

certn box1, all box1, certn consignment,
*certn box1 * (Content2, ceramics),*

where the last expression is built with the help of both the rules P[0], P[1] and the rule with the number 4, the symbol *'certn'* is to be interpreted as the informational item corresponding to the word "certain" in cases when this word is associated with singular.

The rule P[2] allows for constructing the strings of the form $f(a_1,..., a_n)$, where f is a designation of a function, $n>=1$, $a_1,..., a_n$ are l-formulas built with the help of any rules from the list P[0],..., P[10]. The examples of l-formulas built with the help of P[2]:

Distance(Moscow, Tokyo),
*Weight(certn box1 * (Colour, green)(Content2, ceramics))* .

Using the rule P[3], we can build the strings of the form *(a1 ≡ a2)*, where *a1* and *a2* are l-formulas formed with the help of any rules from P[0],..., P[10], and *a1* and *a2* represent the entities being homogeneous in some sense. Examples of K-strings for P[3]:

(Distance(Moscow, Tokyo) ≡ x1), (y1 ≡ y3) ,
(Weight(certn box1) ≡ <8, kg>) .

The rule P[4] is destined, in particular, for constructing K-strings of the form $rel(a_1,..., a_n)$, where *rel* is a designation of n-ary relation, $n>=1$, $a_1,..., a_n$ are the K-strings formed with the aid of some rules from P[0],..., P[10]. The examples of K-strings for P[4]:

Belong(Namur, Cities(Belgium)),
Subset(Cities(Belgium), Cities(Europe)).

The rule P[5] enables us to construct the K-strings of the form *Expr: v* , where *Expr* is a K-string not including *v, v* is a variable, and some other conditions are satisfied. Using P[5], one can mark by variables in the SR of any NL-text: (a) the descriptions of diverse entities mentioned in the text (physical objects, events, concepts, etc.), (b) the SRs of sentences and of larger texts' fragments to which a reference is given in any part of a text. Examples of K-strings for P[5]: *certn box1 : x3 , Higher(certn box1 : x3, certn box1 : x5) : P1 .* The rule P[5] provides the possibility to form SRs of texts in such a manner that these SRs reflect the referential structure of NL-texts. The examples illustrating this property are considered in next sections.

The rule P[6] provides the possibility to build the K-strings of the form ¬ *Expr* , where *Expr* is a K-string satisfying a number of conditions. The examples of K-strings for P[6]:

¬ ship , ¬ personal-communication , ¬ Belong(Osaka, Cities(Belgium)) .

Using the rule P[7], one can build the K-strings of the forms $(a_1 \wedge a_2 \wedge \ldots \wedge a_n)$ or $(a_1 \vee a_2 \vee \ldots \vee a_n)$, where $n>1$, $a_1,...., a_n$ are K-strings designating the entities which are homogeneous in some sense. In particular, $a_1,..., a_n$ may be SRs of assertions (or propositions), descriptions of physical things, descriptions of sets consisting of things of the same kind, descriptions of concepts. The following strings are examples of K-strings (or l-formulas) for P[7]:

(Finland ∨ Norway ∨ Sweden),
(Belong((Namur ∧ Leuven ∧ Ghent), Cities(Belgium)) ∧
¬ Belong(Bonn, Cities((Finland ∨ Norway ∨ Sweden)))).

The rule P[8] allows us to build, in particular, K-strings of the form $c * (rel_1, val_1),..., (rel_n,val_n)$, where *c* is an informational item from the primary universe *X* designating a concept, for $i=1,...,n$, rel_i is the name of a function with one argument or of a binary relation, val_i designates a possible value of rel_i for objects characterized by the concept *c*. The following expressions are examples of K-strings for P[8]:

*box1 * (Content2,ceramics) , firm1 * (Fields, chemistry) ,*
*consignment * (Quantity, 12)(Compos1, box1 * (Content2, ceramics))* .

The rule P[9] permits to build, in particular, the K-strings of the forms $\forall$ *v (conc) D* and $\exists$ *v (conc) D*, where $\forall$ is the universal quantifier, $\exists$ is the existential quantifier, *conc* and *D* are K-strings, *conc* is a designation of a primary concept ("person", "city", "integer", etc.) or of a compound concept ("integer greater than 200", etc.). *D* may be interpreted as a SR of an assertion with the variable *v* about any entity qualified by the concept *conc*. The examples of K-strings for P[9] are as follows: $\forall$*n1 (integer)* $\exists$*n2 (integer) Less(n1,n2),* $\exists$*y (country * (Location, Europe)) Greater(Quantity (Cities (y)),15).*

The rule P[10] is destined for constructing, in particular, the K-strings of the form $<a_1,..., a_n>$, where $n>1$, $a_1,..., a_n$ are K-strings. The strings obtained with the help of P[10] at the last step of inference are interpreted as designations of n-tuples. The components of such n-tuples may be not only designations of numbers, things, but also SRs of assertions, designations of sets, concepts, etc. Using jointly P[10] and P[4], we can build the K-strings

*<8, kg> , Work1(<Agent1, certn man * (First-name,'Ulrich')(Surname, 'Stein')>,*
<Organization, Siemens>,<Start-time, 1996>),

where the thematic roles *Agent1, Organization, Start-time* are explicitly indicated.

4 Restricted Standard K-Languages as a Means for Building Ontologies

The scheme set forth above gives only a very simplified impression about a thoroughly elaborated new mathematical theory including, in particular, the definitions of the class of restricted K-calculuses and the class of restricted standard K-languages (RSK-languages) in simplified conceptual bases(s.c.b.).

Consider some important new opportunities of building ontologies provided by RSK-languages. The examples pertain to biology, ecology, medicine, and business. Suppose that E is a NL-expression, *Semr* is a string of the RSK-language in some s.c.b. and, besides, *Semr* is a possible SR of E. Then we'll say that *Semr* is a K-representation (KR) of E.

Example 1. If D1 = "Freight forward is a freight to be paid in the port of destination" then D1 may have a KR of the form

*(freight-forward ≡ freight * (Description, <x1, Payment-at(x1,*
*certn port1 * (Destination-of, x1))>))* .

The element *certn* is interpreted as the informational item corresponding to the word "certain".

Example 2. If D2 = "Albumin - protein found in blood plasma" then D2 may have a KR of the form

*(albumin ≡ protein * (Location, certn plasma1 * (Location, certn blood))).*

Example 3. Consider the definition D3 = "A flock is a large number of birds or mammals (e.g. sheep or goats), usually gathered together for a definite purpose , such as feeding, migration, or defence". D3 may have the KR

*Definition1 (flock, dynamic-group * (Compos1, (bird ∨ mammal * (Examples, (sheep ∧ goal))), S1, (Estimation1(Quantity(S1), high) ∧ Goal-of-forming (S1, certn purpose * (Examples, (feeding ∨ migration ∨ defence))))).*

Here the string *Compos1* designates the binary relation "Qualitative composition of a set", it connects a set with a concept qualifying each element of this set.

Example 4. The definition D3 is taken from "Longman Dictionary of Scientifc Usage" published in Moscow in 1989. RSK-languages alow for representing definitions and other knowledge items in the object-oriented form, i.e. as an expression reflecting not only the content of the knowledge item but also the values of its external characteristics: the source, the date of publishing, etc. For instance, the information about D3 can be represented as the K-string

*certn inform-object * (Kind, definition)(Content1, Expr1)(source1, certn dictionary * (Title, 'Longman Dictionary of Scientifc Usage')(Publishing-house, (Longman-Group-Limited/Harlow ∧ Russky-Yazyk-Publishers/Moscow)) (City, Moscow)(Year, 1989))* ,

where *Expr1* is the KR of D3 constructed above.

Example 5. Let T1 = "All granulocytes are polymorphonuclear; that is, they have multilobed nuclei". Then T1 may have the following KR:

(Property (arbitr granulocyte : x1, polymorphonuclear) : P1 ∧ Explanation(P1, Follows (Have1(<Agent1, x1>,<Object1, arbitr nucleus : x2>), Property(x2, multilobed)))) .

Here *P1* is the variable marking the meaning of the first phrase of T1; the strings *Agent1, Object1* designate thematic roles (or conceptual cases).

Example 6. Let D4 = "Type AB blood group - persons who possess types A and B isoantigens on red blood cells and no agglutinin in plasma". Then the following formula may be interpreted as a KR of D4:

*Definition2 (type-AB-blood-group, certn set * (Compos1, person) : S1, ∀x1(person) If-and-only-if(Belong1(x1, S1), (Have2(<Agent1, x1>, <Object1, (certn set * (Compos1, type-A-isoantigen) ∧ certn set * (Compos1, type-B-isoantigen))>, <Location, certn set * (Compos1, cell1 * (Part, certn red-blood **

(Belong2, x1)))>) $\wedge$ $\neg$ *Have2(<Agent1, x1>, <Object1, certn set * (Compos1, agglutinin)>, <Location, certn set * (Compos1, plasma1 * (Belong2, x1))>))))* .

Here $\forall$ is the universal quantifier, the string *certn* is interpreted as the informational item corresponding to the word "certain".

Example 7. RSK-languages allow us to build compound designations of various entities. For instance, the sea port Murmansk in the north-west of Russia can be designated by the K-string

*certn port1 * (Title, "Murmansk")*

or by the K-string *certn port1 * (Title, "Murmansk") : v* , where *v* is a variable marking just this particular entity.

The considered examples show such precious opportunities of RSK-languages as building compound designations of concepts and various objects, constructing SRs of definitions (in particular, definitions of concepts qualifying sets and definitions indicating the goals of living beings), reflecting the external connections of definitions, representing SMs of discourses with the references to the meanings of fragments being phrases or larger parts of texts, joining the designations of concepts by means of logical connectives "and", "or". Many additional examples illustrating the possibilities of building ontologies under the framework of the KCL-theory can be found in [11, 12].

5 Some Possibilities of Employing K-strings in the Process of Finding an Answer to a Question of the End User of a Database

The analysis of the expressive possibilities of the formulas of K-calculuses (K-strings) shows that it is possible to construct such a full-text database Db1 that it possesses the properties demonstrated below.

Imagine that there is a big city D., a user of Db1 inputs the question Qs = "Is it true that the ecological situation in the city D. is getting better during the year?", and the date of inputting Qs is Date1. Then Qs is transformed into the following initial K-representation *Semrqs1*:

Question(u1, (u1 $\equiv$ *Truth-value(Better(Ecology(certn city * (Name1, 'D.') : y1, Year(Date1)), Ecology(y1, Last-year(Date1))))))* .

In the expression *Semrqs1*, the element *Ecology* is to be interpreted as the name of the function assigning to the space object *z1* and the year *z2* a statement about the ecological situation in *z1* corresponding to *z2*.

Let's assume that Db1 has the knowledge base Kb1 including a part Objects-list, and this part contains the K-string *certn city * (Name1, 'D.') : v315*. The means that the city D. is associated with the variable *v315*, playing the role of the unique system name of this city. Suppose also that Date1 corresponds to the year 2002. Then *Semrqs1* is transformed into the secondary KR *Semrqs2* of the form

*Question(u1, (u1 ≡ Truth-value(Better(Ecology(certn city * (Name1, 'D.') : v315, 2002), Ecology(v315, 2001))))) .*

Suppose that there is the newspaper "D. News", and one of its issues published in the same month as Date1 contains the following fragment Fr1: *"The quantity of species of birds who made their nests in the city has reached the number 7 in comparison with the number 5 a year ago. It was announced at the press-conference by Mr. Paul Jones, Chair of the D. Association of Colleges Presidents".*

Let's consider a possible way of extracting from this fragment the information for formulation of an answer to Qs. The first sentence Sent1 of Fr1 may have the following KR *Semr1a*:

*((Quantity(certn species * (Compos1, bird)(Descr, <S1, P1>)) ≡ 7) ∧ (P1 ≡ ∃y1(bird)(Element(y1, S1) ∧ ∃e1 (sit) Is (e1, nesting * (Agent1, y1)(Loc, x1)(Time, 2002)))) ∧ ((Quantity(certn species * (Compos1, bird)(Descr, <S2, P2>)) ≡ 5) ∧ (P2 ≡ ∃y2 (bird)(Element(y2, S2) ∧ ∃e2 (sit) Is (e2, nesting * (Agent1, y2)(Loc, x1)(Time,2001))))) : P3 .*

The symbol *certn* is the informational unit corresponding to the word "certain"; *Compos1* is the designation of the binary relation "Qualitative composition of a set"; *P1, P2, P3* are such variables that their type is the distinguished sort "sense of a statement".

Suppose that the second sentence Sent 2 of Fr1 has the following KR *Semr2a:*

*∃e3 (sit) (Is (e3, announcing * (Agent1, x2)(Content1, P3)(Event, certn press-conf : x3)) ∧ (x2 ≡ certn man *(First-name, 'Paul')(Surname, 'Jones')) ∧ (x2 ≡ Chair (certn association1 * (Compos1, scholar * (Be, President (any college * (Loc, certn city * (Name1, 'D.') : x4))))))) .*

Here the element *association1* denotes the concept "association consisting of people" (there are also the associations of universities, cities, etc.).

The analysis of the first sentence Sent1 shows that it is impossible to find directly in Sent 1 the information determining the referent of the word "the city". In this case, let's take into account the knowledge about the source containing the fragment Fr1 and about the use of this knowledge for clarifying the referential structure of published discourses.

Imagine that the knowledge base Kb1 of the considered hypothetical intelligent system contains the K-string

$$Follows\,((\,z1 \equiv \,arbitr\;edition * (Title,\, z2)(Content1,\, Cont1)) \wedge Associated\,(z2,\, arbitr\;space\text{-}object : z3) \wedge Element(w,\, pos,\, Cont1) \wedge Sem\text{-}class(w,\, pos,\, space\text{-}object) \wedge No\text{-}show\text{-}referent\,(w,\, pos,\, Cont1)\,,\, Referent\,(w,\, pos,\, Cont1,\, z3)\,)\,.$$

Let's suppose that: (1) an arbitrary edition *z1* has the title z2 and the content *Cont1*, its title *z2* is associated in any way with the space-object *z3*; (2) the K-string *Cont1* contains the element *w* in the position *pos*, its semantic class is *space-object* (a city, a province, a country, etc.), (3) the text contains no explicit information about the referent of the element *w* in the position *pos* of the formula *Cont1*. Then the argument of the function is the entity denoted by *z3*.

In order to use this knowledge item for the analysis of the fragment Fr1, let's remember that the list of the objects Objects-list (being a part of the knowledge base Kb1) includes the K-string *certn city * (Name1, 'D.') : v315*. Then the system transforms the KR *Semr1a* of the first sentence Sent1 into the formula

$$Semr1b = (Semr1a \wedge (x1 \equiv v315))\;.$$

This means that at this stage of the analysis the information extracted from Sent1 is associated with the city D.

Assume that the knowledge base Kb1 contains the knowledge items

$$\forall z1(person)\; \forall c1(concept)\; Follows(Head(z1,\; arbitr\; association1 * (Compos1,\, c1)),\, Is(z1,\, c1))\,,$$

$$\forall z1(person)\; Follows\,((\,z1 \equiv President((arbitr\;univ : z2\;\; \vee arbitr\;college : z3))),\; Qualification(z1,\, Ph.D.))\,,$$

and these items are interpreted as follows: (1) if a person *z1* is the head of an association of the type 1 (associations consisting of people), the concept *c1* qualifies each element of this association then *z1* is associated with *c1* too; (2) if a person *z1* is the president of a university or a college then *z1* has at least a Ph.D. degree.

Proceeding from the indicated knowledge items and from Semr2a, the system builds $Semr2b = (Semr2a \wedge (x1 \equiv v315))$ and then infers the formula *Qualification(x2, Ph.D.),* where the variable *x2* denotes Mr. Paul Jones, Chair of the D. Association of Colleges Presidents.

Let Kb1 contain also the K-string

$$Follows\,(\exists e1(sit)\; Is(e1,\, announcing * (Agent1,\, arbitr\; scholar * (Qualif,\, Ph.D.))(Kind\text{-}of\text{-}event,\, \neg\, personal\text{-}communication)(Content1,\, Q1)(Time,\, t1)),\; Truth\text{-}estimation(Q1,\, t1,\, <0.9,\, 1>))\;\;,$$

interpreted as follows: if a scholar having a Ph.D. degree announces something, and it is not a personal communication then the estimation of the truth of the announced information has a value in the interval [0.9, 1.0]. Here the substring *∃ e1(sit) Is(e1, announcing ** is to be read as "There is an event *e1* of the type "announcing" such that".

So proceeding from the KRs Semr1b and Semr2b (the secondary KRs of the first and second sentences of the fragment Fr1) and the mentioned knowledge items from Kb1, the system infers the K-string

$$((Quantity(certn\ species * (Compos1, bird)(Descr, <S1, P1>)) \equiv 7) \wedge (P1 \equiv \exists y1 (bird)(Element(y1, S1) \wedge \exists e1\ (sit)\ Is\ (e1, nesting * (Agent1, y1)(Loc, v315)(Time, 2002))) \wedge (Quantity(certn\ species * (Compos1, bird)(Descr, <S2, P2>)) \equiv 5) \wedge (P2 \equiv \exists y2(bird)(Element(y2, S2) \wedge \exists e2\ (sit)\ Is\ (e2, nesting * (Agent1, y2)(Loc, v315)(Time,2001))))) .$$

Suppose that Kb1 contains the following knowledge items:

$$\forall z1(space\text{-}object)\ \forall t1(year)\ Follows\ (Better(Ecolog\text{-}sit(z1, bird, t1), Ecolog\text{-}sit(z1, bird, t2)), Better(Ecology\ (z1, t1), Ecology\ (z1, t2))) ,$$

$$\forall t1(year)\ \forall t2(year)\ Follows(((Set1 \equiv certn\ species * (Compos1, bird)(Descript, Q1)) \wedge (Q1 \equiv \exists y1(bird)(Element(y1, Set1) \wedge \exists e1\ (sit)\ Is\ (e1, nesting * (Agent1, y1)(Loc,x1)(Time, t1)))) \wedge (Set2 \equiv certn\ species * (Compos1, bird)(Descript, Q2)) \wedge Analogue\ (Q2, Q1, < Previous\text{-}year\ (t1), t1>) \wedge Greater\ (Quantity(Set1), Quantity(Set2))), Better(Ecolog\text{-}sit(z1, bird, t1), Ecolog\text{-}sit(z1, bird, Previous\text{-}year(t1)))) .$$

Here the substring *Analogue (Q2, Q1, < Previous-year (t1), t1>)* means the K-string marked by the variable *Q2* can be obtained from the K-string *Q1* by means of replacing the variable *t1* in *Q1* with the string *Previous-year (t1).*

That is why the system finally infers the formulas

$$Better(Ecolog\text{-}sit(v315, bird, 2002), Ecolog\text{-}sit(v315, bird, 2001)) ,$$

$$Better(Ecology(v315, 2002), Ecology(v315,2001)) .$$

Hence the system formulates the answer “YES” and adds the expression of the form *Ground: Fr1 (‘D. News’, Date1).*

The considered examples show that RSK-languages enable us, in particular, to describe the conceptual structure of texts with: (a) references to the meanings of phrases and larger parts of texts , (b) compound designations of sets, (c) the names of the functions determined on sets, (d) complicated designations of objects , (e) generalized quantifiers ("arbitrary", "certain", etc.). Besides, RSK-languages provide the possibilities to build formal analogues of compound concepts, to describe the semantic structure of definitions, to mark by variables the designations of objects and sets of objects, to reflect thematic roles, to represent linguistic metaknowledge.

The advantages of the RKCL-theory in comparison with Discourse Representation Theory [4, 5] and Episodic Logic [8-10] are, in particular, the possibilities: (1) to distinguish in a formal way objects (physical things, events, etc.) and concepts qualifying these objects; (2) to build compound representations of concepts; (3) to distinguish in a formal manner objects and sets of objects, concepts and sets of concepts; (4) to build complicated representations of sets, sets of sets, etc.; (5) to describe set-theoretical relationships; (6) to describe effectively structured meanings (SMs) of discourses with references to the meanings of phrases and larger parts of discourses; (7) to describe SMs of sentences with the words "concept", "notion"; (8) to describe SMs of sentences where the logical connective "and" or "or" joins not the expressions-assertions but designations of things, sets, or concepts; (9) to build complicated designations of objects and sets; (10) to consider non-traditional functions with arguments or/and values being sets of objects, of concepts, of texts' semantic representations, etc.; (11) to construct formal analogues of the meanings of infinitives with dependent words.

The items (3) – (8), (10), (11) indicate the principal advantages of the KCL-theory in comparison with the Theory of Conceptual Graphs [6, 7]. Besides, the expressive possibilities of the KCL-theory are much higher than the possibilities of TCG as concerns the items (1), (2), (9).

6 Conclusions

Since the middle of the nineties, the KCL-theory has been successfully applied to the design of several NL-processing systems: Russian-language interfaces to relational databases (DB); computer systems fulfilling information retrieval in Russian texts stored in a full-text DB or conceptual processing of the discourses in English being available on-line due to the World Wide Web [13]; a NL-interface to an autonomous intelligent robot working at an airport (this NL-interface is a part of an animation system but not of a real robot) [14].

It appears that the KCL-theory may essentially contribute to the speeding-up of the progress in the theory and practice of (a) conceptual information retrieval in textual databases, (b) automatic text summarization, (c) automatic extraction of information from NL-texts for forming and up-dating knowledge bases of applied intelligent systems (AIS), (d) design of more flexible and powerful NL-interfaces to AIS, (e) building ontologies of arbitrary application domains, (f) elaborating content languages for multi-agent systems.

Acknowledgements. I am grateful to three anonymous referees of the initial version of this paper for their work and remarks helped to improve the presentation style of the paper.

References

1. Fomichov, V.A.: Integral Formal Semantics and the Design of Legal Full-Text Databases. Cybernetica (Belgium), XXXVII (1994), 145-177
2. Fomichov, V.A.: A Variant of a Universal Metagrammar of Conceptual Structures. Algebraic Systems of Conceptual Syntax. In Nijholt, A., Scollo, G., Steetskamp, R. (eds.), Algebraic Methods in Language Processing. Proceedings of the Tenth Twente Workshop on Language Technology joint with First AMAST Workshop on Language Processing, University of Twente, Enschede, The Netherlands, Dec. 6 – 8 (1995) 195 - 210

3. Fomichov, V.A.: A Mathematical Model for Describing Structured Items of Conceptual Level. Informatica (Slovenia). 20 (1996) 5-32
4. van Eijck, D.J.N., Kamp, H.: Representing Discourse in Context. Amsterdam, The University of Amsterdam (1996)
5. Kamp, H.,Reyle, U.: A Calculus for First Order Discourse Representation Structures. Journal for Logic, Language and Information, 5 (1996) 297-348.
6. Sowa, J.F.: Towards the Expressive Power of Natural Languages. In: Sowa, J.F. (ed.): Principles of Semantic Networks. Morgan Kaufmann Publishers, San Mateo, CA (1991) 157-189.
7. Sowa, J.F.: Conceptual Graphs: Draft Proposed American National Standard. In: Tepfenhart, W., Cyre, W. (eds.): Conceptual Structures: Standards and Practices. Lecture Notes in AI, Vol.1640. Springer-Verlag, Berlin Heidelberg New York (1999) 1-65.
8. Allen, J.F., Schubert, L. K., Ferguson, G. M., Heeman, P. A., Hwang, C. H., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., Traum, D.R.: The TRAINS Project: A Case Study in Building a Conversational Planning Agent. J. Experimental and Theoretical Artificial Intelligence 7 (1995) 7-48
9. Hwang, C.H., Schubert, L.K.: Episodic Logic: A Situational Logic for Natural Language Processing. In: Aczel, P., Israel, D., Katagiri, Y., Peters, S. (eds.): Situation Theory and Its Applications, Vol. 3. CSLI, Stanford (1993) 307-452
10. Schubert, L.K., Hwang, C.H.: Episodic Logic Meets Little Red Riding Hood: A Comprehensive, Natural Representation for Language Understanding. In: Iwanska, L., Shapiro, S.C (eds.): Natural Language Processing and Knowledge Representation: Language for Knowledge and Knowledge for Language. MIT/AAAI Press, Menlo Park, CA, and Cambridge, MA (2000) 111-174
11. Fomichov, V.A.: K-calculuses and K-languages as Powerful Formal Means to Design Intelligent Systems Processing Medical Texts. Cybernetica. XXXVI (1993) 161-182
12. Fomichov, V.A.: An Ontological Mathematical Framework for Electronic Commerce and Semantically-structured Web. In:.Zhang, Y., Fomichov, V.A., Zeleznikar, A.P. (eds.): Special Issue on Database, Web, and Cooperative Systems. Informatica (Slovenia). 24 (2000) 39-49
13. Fomichov, V.A., Kochanov, A.A.: Principles of Semantic Search for Information on the Web by the Intelligent Agent LingSearch-1. In: Pohl, J., Fowler IV, T. (eds.): Proceedings of the Focus Symposia on Advances in Computer-Based and Web-Based Collaborative Systems; InterSymp-2001, the 13th International Conference on Systems Research, Informatics and Cybernetics, July 31 – August 1, 2001, Baden-Baden, Germany. Collaborative Agent Design (CAD) Research Center, Cal Poly, San Luis Obispo, CA, USA (2001) 121 – 131
14. Fomichov, V.A.: The Method of Constructing the Linguistic Processor of the Animation System AVIAROBOT. In: Pohl, J. (ed.): Proceedings of the Focus Symposium on Collaborative Decision-Support Systems; InterSymp-2002, the 14th International Conference on Systems Research, Informatics and Cybernetics, July 29 – August 3, 2002, Baden-Baden, Germany. CAD Research Center, Cal Poly, San Luis Obispo, CA, USA (2002) 91 – 102

X-Compass: An XML Agent for Supporting User Navigation on the Web

Salvatore Garruzzo, Stefano Modafferi, Domenico Rosaci, and Domenico Ursino

DIMET - Università "Mediterranea" di Reggio Calabria
Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy
{garruzzo,stmod}@tiscali.it,{rosaci,ursino}@ing.unirc.it

Abstract. In this paper we present X-Compass, an XML agent for supporting a user during her/his navigation on the Web. This agent is the result of our attempt of synthesizing, in a unique context, important guidelines currently characterizing the research in various Computer Science sectors. X-Compass constructs and handles a rather rich, even if light, user profile. This latter is, then, exploited for supporting the user in the efficient search of information of her/his interest; in this way, the proposed agent behaves as a content-based recommender system. Moreover, X-Compass is particularly suited for constructing multi-agent systems and, therefore, for implementing collaborative filtering recommendation techniques. In addition, being based on XML, X-Compass is particularly light and capable of operating on various hardware and software platforms. Finally, the exploitation of XML makes the information exchange among X-Compass agents and, therefore, the management and the exploitation of X-Compass multi-agent systems, easy.

1 Introduction

In the last years the enormous development of new technologies caused an exponential growth of available information; while in the past the most important problem to solve was the lack of desired information, presently the greatest difficulty to face is the efficient management of a great amount of data. In order to suitably carry out such a task, the necessity arises of either automatic or semi-automatic tools capable of supporting a user in the efficient search of information of her/his interest. In order to provide the required support, these tools should "know" the user; this capability can be gained by constructing and handling a user profile. In the literature various approaches have been proposed for carrying out such a task. Some of them require the support of the user; they produce rather simple profiles which, owing to their extreme lightness, can be easily managed and exploited [5,16]. Some other approaches extract user profiles by applying data mining techniques on information about user past behaviour; in this way, they obtain rather rich profiles; however, a great amount of work is necessary for constructing and managing them [2,13]. Finally, another group of approaches constructs a user profile as a synthesis of the information sources visited by the user in the past; in order to perform their activity, they exploit information source integration techniques. In this way, obtained profiles are particularly rich and complex; however, the required activity for constructing and handling them is particularly heavy [14].

After a profile has been obtained, it can be exploited for providing the user with personalized recommendations. In the literature various *Recommender Systems* have been proposed [1]; the approaches they follow can be sub-divided into two categories, named content-based and collaborative filtering. *Content-based* approaches recommend to a user those resources appearing to be similar to other ones already accessed by her/him in the past [17]. *Collaborative filtering* methodologies recommend to a user those resources accessed in the past by other users having a similar profile [17,19].

The attempt to synthesize, in a unique context, the three activities described above (i.e., the construction of a user profile, its exploitation for supporting the user in the efficient search of information of her/his interest and the management of collaboration among users) has been carried out in the past by exploiting intelligent agents. Firstly, *software agents* have been proposed; these merge the capability of constructing a user profile, by analyzing user past behaviour, with the capability, typical of content-based recommender systems, of exploiting such a profile for supporting the user in her/his activities [9]. After this, some multi-agent systems have been proposed which not only carry out the two activities described above but also cooperate by exchanging information about users; this cooperation is exploited for recommending to their users the next information sources to visit [15].

Cooperation in multi-agent systems requires a continuous information exchange among involved agents. In the past, exchanged information was quite simple and easily manageable in such a way that the management cost, required to involved agents, was quite low. Nowadays, data exchange has become a central problem to face in the field of Web-based information systems. The most promising solution to this problem was the definition of XML, a novel language for representing and exchanging data over the Internet. XML embodies both representation capabilities, typical of HTML, and data management features, typical of DBMS's; it is presently considered as a standard for the Web. XML capabilities make it particularly suited to be exploited in the agent research; as a matter of fact, agents using XML for both representing and handling their own ontology have been proposed, especially in the e-commerce sector [8,20]; the ontology handled by such systems is generally quite light and based on is-a relationships. In addition, proposed agents provide no mechanism that, at the same time, constructs, handles and exploits a user profile for providing both content-based and collaborative filtering recommendations.

This paper deals with the issues relative to this scenario. In particular, it describes X-Compass, an agent designed for facing all requirements illustrated above.

Firstly X-Compass constructs and handles a user profile representing its ontology; profile construction and management is carried out automatically, by monitoring the behaviour of a user during her/his accesses to information sources of her/his interest. Such a profile represents user interests, as well as some relationships existing among them. Two different kinds of relationships are handled, namely: *(i) is-a relationships*, organizing user interests in a generalization hierarchy; *(ii) associative relationships*, linking user interests appearing distant in the is-a hierarchy but determined to be close from the analysis of the user behaviour; the extraction of these relationships requires the exploitation of data mining techniques.

After the user profile has been constructed, X-Compass exploits it for carrying out both content-based and collaborative filtering recommendation activities. X-Compass content-based recommendation activities consist of: *(i)* suggesting to the user the next information source to visit, on the basis of both the content of the current information source and user profile; *(ii)* providing a semantic support to existing Web search engines in such a way to make them capable to adapt their results to the user profile. X-Compass collaborative filtering recommendation activities are strongly based on both the capability of our agent of easily communicating with other X-Compass agents and the consequent attitude of easily creating and handling multi-agent systems. In this paper we present two X-Compass multi-agent systems; the former is devoted to handle knowledge sharing among users, the latter is able to predict future interests of a user by analyzing the past behaviour of other users having similar profiles.

A further interesting characteristic of X-Compass consists of exploiting XML for both storing and handling its knowledge base, i.e., both its ontology and data structures supporting its activities. The exploitation of XML, instead of a traditional DBMS, makes X-Compass well suited for operating on various, both hardware and software, platforms, ranging from workstations to personal computers, notebooks and palmtops. XML exploitation appears to be even more interesting for multi-agent system management since one of the most interesting XML features consists of its capability to support data exchange.

All considerations illustrated above allow to conclude that, even if X-Compass handles quite a rich user profile and can be exploited in a large range of applications, it is particularly light and versatile, capable of operating in a great variety of application contexts.

The plan of the paper is as follows: Section 2 describes the X-Compass knowledge base. Section 3 illustrates the X-Compass general behaviour; in particular, it describes both the construction and the management of a user profile. The use of X-Compass for providing content-based recommendations is presented in Section 4 whereas Section 5 describes X-Compass exploitation in collaborative filtering recommendation activities. Finally, in Section 6, we draw our conclusions.

2 The X-Compass Knowledge Base

The knowledge base associated with an X-Compass agent consists of both its ontology and the set of data structures supporting its activities. It has been realized in XML; however, in a first time, for making its comprehension easier, it shall be described in terms of abstract data structures, such as graphs and lists. In order to describe it, we shall refer to both a user u and the X-Compass agent $Ag_{xc}(u)$ associated with u.

The support knowledge base of $Ag_{xc}(u)$ consists of the following triplet:

$$\langle P(u), H(u), AH(u) \rangle$$

In the following we shall describe each of these components.

The User Profile P(u). $P(u)$ denotes the profile of u, i.e., it stores both the interests and the preferences of u. It is a rooted graph:

$$P(u) = \langle NS(u), AS(u) \rangle = \langle NS(u), AS_I(u) \cup AS_A(u) \rangle$$

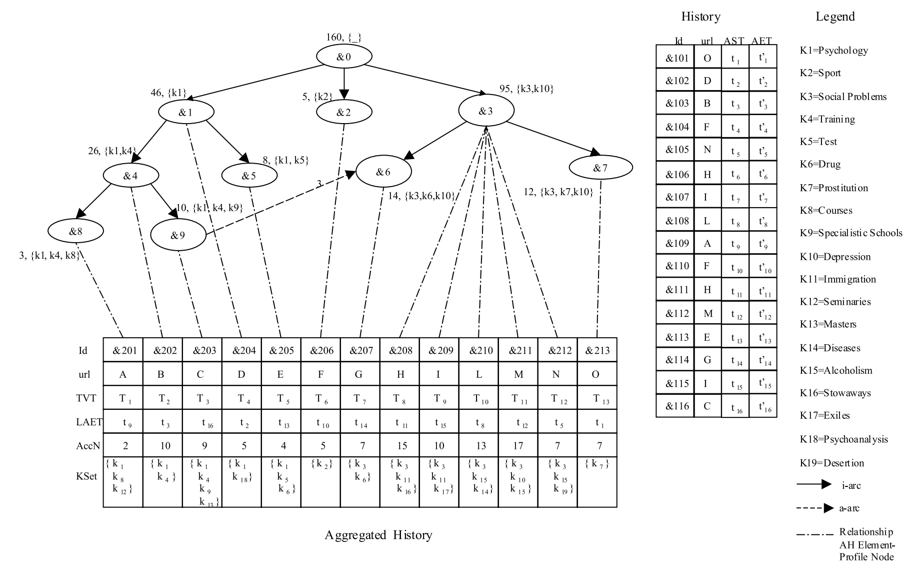
$NS(u)$ represents the set of nodes of $P(u)$; each of them, with the exception of the root, denotes an *interest* of u. A node $N_i \in NS(u)$ is characterized by: *(i)* an *Identifier* Id_{N_i}; *(ii)* an *Attractiveness Degree* $DAttr_{N_i}$; *(iii)* a *Keyword Set* $KSet_{N_i}$; these latter, as a whole, define the semantics of the interest associated with N_i. Since an interest corresponds to each node, and vice versa, in the following we shall exploit the terms "node" and "interest" interchangeably.

$AS(u)$ denotes the set of arcs of $P(u)$. It is composed of the two subsets $AS_I(u)$ and $AS_A(u)$. $AS_I(u)$ is the set of *i-arcs*. An i-arc $A_i = \langle A, B \rangle \in AS_I(u)$ indicates that the interest B is a specialization of the interest A. $AS_A(u)$ is the set of *a-arcs*. Each a-arc $\langle A, B, L_{AB} \rangle \in AS_A(u)$ denotes the existence of an association rule of the form $A \rightarrow B$ [3]. A label L_{AB} is associated with each a-arc; it represents an *Aging Coefficient* whose semantics and role shall be explained in the following.

Given a User Profile $P(u)$, the *Profile Tree* $PrTree(u) = \langle NS(u), AS_I(u) \rangle$ is defined as the tree composed by nodes and i-arcs of $P(u)$. □

The "History" H(u). $H(u)$ represents an ordered list of elements, each associated with a Web page access of u; $H(u)$ order reflects the temporal access succession. Each element $h_i \in H(u)$ is characterized by: *(i)* an *Identifier* Id_{h_i}; *(ii)* a *URL* url_{h_i}; *(iii)* an *Access Starting Time* AST_{h_i}, denoting the starting time instant of the access represented by h_i; *(iv)* an *Access Ending Time* AET_{h_i}, denoting the ending time instant of the access associated with h_i. $H(u)$ is exploited for updating $AS_A(u)$; it is, therefore, a dynamic structure and all its elements are deleted at the end of each $AS_A(u)$ update. □

The "Aggregated History" AH(u). $AH(u)$ is a list of elements, each associated with a Web page. An element $ah_i \in AH(u)$ is characterized by: *(i)* an *Identifier* Id_{ah_i}; *(ii)* a *URL* url_{ah_i}; *(iii)* a *Total Visit Time* TVT_{ah_i}, representing the total time which u spent for visiting the page associated with ah_i, possibly during different accesses ; *(iv)* a *Last Access Ending Time* $LAET_{ah_i}$, representing the ending time instant of the last access of u to the page associated with ah_i; *(v)* the *Access Number* $AccN_{ah_i}$ of u relative to the page corresponding to ah_i; *(vi)* a *Keyword Set* $KSet_{ah_i}$; these keywords, in the whole, define the semantics of the page associated with ah_i. A unique node N_{ah_i} of $P(u)$ is associated with ah_i; N_{ah_i} is called *Representative Node* of ah_i in $P(u)$. $AH(u)$ is exploited for



History

Id	url	AST	AET
&101	O	t_1	t'_1
&102	D	t_2	t'_2
&103	B	t_3	t'_3
&104	F	t_4	t'_4
&105	N	t_5	t'_5
&106	H	t_6	t'_6
&107	I	t_7	t'_7
&108	L	t_8	t'_8
&109	A	t_9	t'_9
&110	F	t_{10}	t'_{10}
&111	H	t_{11}	t'_{11}
&112	M	t_{12}	t'_{12}
&113	E	t_{13}	t'_{13}
&114	G	t_{14}	t'_{14}
&115	I	t_{15}	t'_{15}
&116	C	t_{16}	t'_{16}

Legend

K1=Psychology
K2=Sport
K3=Social Problems
K4=Training
K5=Test
K6=Drug
K7=Prostitution
K8=Courses
K9=Specialistic Schools
K10=Depression
K11=Immigration
K12=Seminaries
K13=Masters
K14=Diseases
K15=Alcoholism
K16=Stowaways
K17=Exiles
K18=Psychoanalysis
K19=Desertion

Id	&201	&202	&203	&204	&205	&206	&207	&208	&209	&210	&211	&212	&213
url	A	B	C	D	E	F	G	H	I	L	M	N	O
TVT	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
LAET	t_9	t_3	t_{16}	t_2	t_{13}	t_{10}	t_{14}	t_{11}	t_{15}	t_8	t_{12}	t_5	t_1
AccN	2	10	9	5	4	5	7	15	10	13	17	7	7
KSet	$\{k_1, k_8, k_{12}\}$	$\{k_1, k_4\}$	$\{k_1, k_4, k_9, k_{13}\}$	$\{k_1, k_{18}\}$	$\{k_1, k_5, k_6\}$	$\{k_2\}$	$\{k_3, k_6\}$	$\{k_3, k_{11}, k_{16}\}$	$\{k_3, k_{11}, k_{17}\}$	$\{k_3, k_{15}, k_{14}\}$	$\{k_3, k_{10}, k_{15}\}$	$\{k_3, k_{15}, k_{19}\}$	$\{k_7\}$

Aggregated History

Fig. 1. The knowledge base associated with an X-Compass agent

updating $NS(u)$ and $AS_I(u)$. It is, therefore, a dynamic structure; in particular, elements denoting a low *Attractiveness Degree* for u (computed starting from TVT_{ah_i}, $LAET_{ah_i}$ and $AccN_{ah_i}$) are removed whenever $NS(u)$ and $AS_I(u)$ are updated. □

From the previous descriptions, it is possible to deduce that $P(u)$ is an intensional structure whereas $H(u)$ and $AH(u)$ are extensional ones; as a consequence, the dynamism of $P(u)$ is far lower than that characterizing $H(u)$ and $AH(u)$.

Figure 1 illustrates the knowledge base of an X-Compass agent associated with a user, say Peter, who is a psychologist. The figure illustrates $P(Peter)$, $H(Peter)$ and $AH(Peter)$; due to layout reasons, in the figure, we have represented each keyword by an identifier. Node &8 of $P(Peter)$ represents an interest; the Attractiveness Degree associated with this node is 3; the associated keywords are {Psychology, Training, Courses}. The i-arc $\langle \&4, \&8 \rangle$ indicates the existence of an is-a relationship from &4 to &8; in particular, it indicates that the interest associated with &8 is a specialization of the interest associated with &4. The a-arc $\langle \&9, \&6, 3 \rangle$ indicates the existence of an association rule &9 $\rightarrow$ &6; the Aging Coefficient of this arc is 3. Finally, a link exists between the element &209 of $AH(Peter)$ and the node &3 of $P(Peter)$ denoting that &3 is the Representative Node of &209 in $P(Peter)$.

As previously pointed out, $P(u)$, $H(u)$ and $AH(u)$ are XML documents. The DTD associated with them is represented in Figure 2. It models exactly their description, as provided above. Links between nodes of $P(u)$ and elements of $AH(u)$ are realized by X-links. In the following, whenever no ambiguity might arise, we shall denote $P(u)$, $H(u)$ and $AH(u)$ simply by P, H and AH.

```
<!ELEMENT Profile (Interest*, Relationship*)>

<!ELEMENT Interest (Keyword*, Link*)>
    <!ATTLIST Interest
       Identifier ID #REQUIRED
       Attractiveness_Degree CDATA #REQUIRED
    >

<!ELEMENT Relationship (#PCDATA)>
    <!ATTLIST Relationship
       Identifier ID #REQUIRED
       Typology (i-arc|a-arc) #REQUIRED
       Aging_Coefficient CDATA #IMPLIED
       From IDREF #REQUIRED
       To IDREF #REQUIRED
    >

<!ELEMENT Keyword (#PCDATA)>

<!ELEMENT Link ANY>
    <!ATTLIST Link
       xmlns:xlink CDATA #FIXED ``http://www.w3.org/1999/xlink''
       xlink:type CDATA #FIXED ``simple''
       xlink:href CDATA #REQUIRED
       INLINE CDATA #FIXED ``true''
    >
```

```
<!ELEMENT History (Element*)>

 <!ELEMENT Element (#PCDATA)>
    <!ATTLIST Element
       Identifier ID #REQUIRED
       url CDATA #REQUIRED
       Access_Starting_Time CDATA #REQUIRED
       Access_Ending_Time CDATA #REQUIRED
       Next IDREF #REQUIRED
    >

 <!ELEMENT Aggregated_History (Aggregated_Element+)>

 <!ELEMENT Aggregated_Element(Keyword+, Link)>
    <!ATTLIST Aggregated_Element
       Identifier ID #REQUIRED
       url CDATA #REQUIRED
       Total_Visit_Time CDATA #REQUIRED
       Last_Access_Ending_Time CDATA #REQUIRED
       Access_Number CDATA #REQUIRED
       Next IDREF #REQUIRED
    >
```

Fig. 2. The DTD of the Knowledge Base associated with an X-Compass agent

3 The X-Compass General Behaviour

In X-Compass, the user activity is modeled as a succession of sessions. Therefore, before describing the agent behaviour, it is necessary to provide a more formal definition of the concept of session.

Definition 1. Let $a_0, \ldots, a_n$ be a succession of Web page accesses carried out by u. Let AST_{a_i} (risp., AET_{a_i}) be the Access Starting Time (resp., Access Ending Time) relative to a_i. The *Latency Time* LT_{a_i} is defined as $AST_{a_i} - AET_{a_{i-1}}$ for each i such that $1 \leq i \leq n$.

A *Micro-Session* is defined as a succession a_0, …, a_h of Web page accesses such that $LT_{a_i} \leq th_{MSess}$, for each i such that $1 \leq i \leq h$. th_{MSess} is a threshold identifying the maximum time interval which can elapse between two accesses of u in the *same Micro-Session*. In our approach $th_{MSess} = 1350$ seconds, according to the experimental results of [6].

A *Session* is defined as a succession of k Micro-Sessions; here, k is a suitably defined integer, greater than or equal to 1[1]. □

We are now able to describe the behaviour of $Ag_{xc}(u)$. During each session, the agent monitors each Web page access carried out by u and extracts the necessary information; then, it updates H, AH and, finally, the Attractiveness Degree of the Representative Node of the currently visited page. After this, it exploits P for providing content-based recommendations (as an example, for suggesting to u the next page to visit). At the end of each session, P, H and AH are updated; the new P is, then, exploited for providing u with content-based or collaborative filtering recommendations. More formally, an X-Compass agent can be modeled as shown in Figure 3.
Procedures and Functions introduced above behave as follows:

– $Create\_Profile\_Root$ initializes P; in particular it creates its root;

[1] Generally k is much greater than 1.

```
The X-Compass Agent
Knowledge Base
    Ontology: the User Profile $P$;
    Support Structures: the History $H$, the Aggregated History $AH$;
Behaviour
    var $KSet$: a keyword set; $w$: a url; $ast, aet$: a time instant; $T_{nidl}$; a time interval;
    begin
        $P := \emptyset$ ; $H := \emptyset$; $AH := \emptyset$;
        $Create\_Profile\_Root()$;
        repeat
            repeat
                $[w, ast, aet, T_{nidl}, KSet] := Monitor\_Access\_Page()$;
                $Add\_History\_Element(w, ast, aet, T_{nidl})$;
                if $Is\_Aggr\_History\_Present(w)$ then $Update\_Aggr\_History\_Element(w, aet, T_{nidl})$
                else $Create\_Aggr\_History\_Element(w, aet, T_{nidl}, KSet)$;
                $Update\_Profile\_Node(w, T_{nidl})$;
                $Recommend\_User()$;
            until not $Verify\_End\_Session()$;
            $Restructure\_Knowledge\_Base()$;
            $Exploit\_Ontology()$;
        until false
    end
```

Fig. 3. The model of an X-Compass Agent

- $Monitor\_Access\_Page$ monitors u during her/his access to a Web page and returns the URL w of the visited page, the Access Starting Time ast, the Access Ending Time aet, the "no-idle" time interval T_{nidl} spent by u during her/his access to the page[2] and the Keyword Set $KSet$ associated with the visited page; $KSet$ can be derived by applying any of the approaches proposed in the literature for extracting the most important keywords associated with a Web page, for instance an approach based on the TFIDF technique [10,18].
- $Is\_Aggr\_History\_Present$ receives a URL w and returns *true* if there exists an element $ah \in AH$ such that $url_{ah} = w$, *false* otherwise.
- $Update\_Aggr\_History\_Element$ receives a URL w, the Access Ending Time aet and the no-idle time interval T_{nidl}; it updates the Aggregated History element ah_w whose URL is w. In particular, $TVT_{ah_w} = TVT_{ah_w} + T_{nidl}$, $LAET_{ah_w} = aet$, $AccN_{ah_w} = AccN_{ah_w} + 1$.
- $Create\_Aggr\_History\_Element$ creates a new element in AH, defines the corresponding parameters and associates it with a suitable node of P, which becomes its Representative Node. This procedure is described into detail in Section 3.1.
- $Update\_Profile\_Node$ receives a URL w and the no-idle time interval T_{nidl} which u spent for visiting the page having w as URL; it updates the Attractiveness Degree of the node N_w representing ah_w in P. In particular, $DAttr_{N_w} = DAttr_{N_w} + 1 + \lfloor \frac{T_{nidl}}{q} \rfloor$. Such a formula denotes that the Attractiveness Degree of a node is directly proportional to both the number of accesses to the Web pages it represents, and the no-idle time interval which u spent during these accesses. The term q used in this formula is necessary for normalizing the no-idle time interval w.r.t. the number of accesses.
- $Recommend\_User$ exploits P for providing u with content-based recommendations (as an example for suggesting to her/him the next page to visit).

[2] It is worth pointing out that T_{nidl} might be less than $aet - ast$; this happens when, during the access, there exist some periods of user inactivity.

Some of the possible content-based recommendation activities are described into detail in Section 4.
- $Verify\_End\_Session$ verifies if the current session is finished; in order to carry out such a task it exploits Definition 1.
- *Restructure_Knowledge_Base* updates P, H and AH. All details about this procedure can be found in Section 3.2.
- *Exploit_Ontology* exploits the updated P for providing u with both content-based and collaborative filtering recommendations. Some of the content-based (resp., collaborative filtering) recommendation activities, which can be carried out with the support of an X-Compass agent, are described in Section 4 (resp., Section 5).

3.1 Procedure Create_Aggr_History_Element

This procedure receives a URL w, an Access Ending Time aet, a no-idle time interval T_{nidl} and a Keyword Set $KSet$. Firstly it creates a new element ah_{new} in AH and sets $url_{ah_{new}} = w$, $TVT_{ah_{new}} = T_{nidl}$, $LAET_{ah_{new}} = aet$, $AccN_{ah_{new}} = 1$ and $KSet_{ah_{new}} = KSet$. Then it determines the Representative Node of ah_{new}. Such an operation is based on keyword comparisons and is carried out with the support of the function $Max\_Weight\_Matching$.

$Max\_Weight\_Matching$ receives two keyword sets KS_i and KS_j; it returns a number $DSim_{ij}$, in the real interval $[0, 1]$, representing the similarity degree between KS_i and KS_j. In order to compute $DSim_{ij}$, $Max\_Weight\_Matching$ first constructs a bipartite graph $BG = (U \cup V, E)$, where U (resp., V) is a set of nodes such that each of them is associated with a keyword of KS_i (resp., KS_j), and E is a set of arcs, such that each of them links a node of U with a node of V; in particular, an arc exists between $u_s \in U$ and $v_t \in V$ if the keywords corresponding to u_s and v_t are synonymous; in order to determine keyword synonymies, a suitable thesaurus, such as Wordnet [12], is exploited. After BG has been constructed, $Max\_Weight\_Matching$ determines the set $E' \subseteq E$ of arcs such that, for each node $x \in U \cup V$, at most one arc of E' is incident onto x and $|E'|$ is maximum. The value $DSim_{ij}$ returned by the function, which corresponds to the value of the objective function associated with the Maximum Weight Matching on BG, is $DSim_{ij} = \frac{2|E'|}{|U|+|V|}$. For more details about the Maximum Weight Matching problem, the interested reader is referred to [7].

We are now able to describe how the Representative Node of ah_{new} is determined. Firstly, the Keyword Set of ah_{new} is first compared with the Keyword Set of each node of the first level of the Profile Tree $PrTree$ corresponding to P. Each comparison, carried out by exploiting the function $Max\_Weight\_Matching$, returns a Similarity Degree between the Web page associated with ah_{new} and a user interest. After this, the set $SimSet$ of nodes of the first level of $PrTree$ whose Similarity Degree with ah_{new} is greater than a certain threshold th_{Sim}, is constructed. If $SimSet$ is empty, ah_{new} is associated with the parent of the nodes into examination (which, in this case, is the root of P). Vice versa, if $SimSet$ is not empty, the node N_{Max}, having the maximum derived Similarity

Degree with ah_{new}, is determined and the search continues by applying the same methodology to the sub-tree of $PrTree$ having N_{Max} as root. Such an iterative process terminates when either no node of the first level of the current sub-tree has a Similarity Degree with ah_{new} greater than th_{Sim} or the current sub-tree is empty (this last case happens when the search process has reached a leaf node of $PrTree$).

3.2 Procedure Restructure_Knowledge_Base

This procedure is activated at the end of each session for restructuring P, H and AH.

Restructuring AH. This task implies to eliminate those elements of AH having a low (i.e., smaller than a certain threshold th_{Attr}) Attractiveness Degree. Given an element $ah \in AH$, its Attractiveness Degree $DAttr_{ah}$ is computed by applying the following formula, which takes into account the Access Number $AccN_{ah}$, the Total Visit Time TVT_{ah} and the Last Access Ending Time $LAET_{ah}$:

$$DAttr_{ah} = AccN_{ah} + \lfloor \frac{TVT_{ah}}{q} \rfloor - \lfloor \frac{CT - LAET_{ah}}{q'} \rfloor$$

Here, CT denotes the *Current Time*, i.e., the instant when the formula is computed (corresponding to the time instant when $Restructure\_Knowledge\_Base$ is activated). This formula indicates that the value of $DAttr_{ah}$ increases with the increase of the Access Number and the Total Visit Time and decreases with the increase of the time interval elapsed from the last access. q (already introduced in Section 3) and q' are suitable normalization values. Note that $DAttr_{ah}$ might also be negative.

Restructuring P. This task implies: *(i)* to determine if u has new interests, *(ii)* to verify if previous interests are no longer attractive for u, *(iii)* to re-compute the association rules and, consequently, a-arcs in P.

Identification of new user interests. In our model, a new interest arises whenever the Attractiveness Degree of a node of P is high, i.e. greater than a certain threshold th'_{Attr}. For each node N_i having these characteristics, the following operations, leading to the *splitting* of N_i and the consequent appearance of new interests, must be carried out:

- The set $EAHSet_{N_i}$ of AH elements represented in P by N_i is identified.
- The elements of $EAHSet_{N_i}$ are grouped into clusters, each composed by elements denoting homogeneous interests. This clustering task is carried out by: *(i)* activating $Max\_Weight\_Matching$ for determining the Similarity Degree $DSim_{ah_s ah_t}$ between each pair of elements (ah_s, ah_t) such that $ah_s, ah_t \in EAHSet_{N_i}$; *(ii)* constructing a Similarity Matrix $SimMat$ whose generic element $SimMat[s,t] = DSim_{ah_s ah_t}$; *(iii)* applying the clustering algorithm PAM [11] on $SimMat$.

- A new node N_{i_j} is created in P for each cluster C_j obtained by PAM. Let $EAHSet_{N_{i_j}}$ be the set of elements of $EAHSet_{N_i}$ which PAM grouped in C_j: N_{i_j} becomes the Representative Node of each element of $EAHSet_{N_{i_j}}$. The Keyword Set $KSet_{N_{i_j}}$ of N_{i_j} is computed as the union of the Keyword Sets of the elements of $EAHSet_{N_{i_j}}$. $DAttr_{N_{i_j}}$ is computed as $DAttr_{N_{i_j}} = \sum_{k=1}^{|EAHSet_{N_{i_j}}|} \left(AccN_{ah_k} + \lfloor \frac{TVT_{ah_k}}{q} \rfloor \right)$ where q is the tuning coefficient defined in Section 3. Finally, an i-arc from N_i to N_{i_j} is created in P.
- The new Keyword Set $KSet_{N_i}$ of N_i is defined by taking only those keywords associated with at least two nodes obtained during the previous step; this choice is justified by the semantics associated with P, which requires the interest represented by N_i to be a generalization of the interests represented by the nodes created during the previous steps. Since, currently, no element of AH is represented in P by N_i, the Attractiveness Degree of N_i is set to 0[3]. Since the semantics of N_i has been deeply modified, all a-arcs relative to it are removed[4].

Removal of the interests no longer attractive for the user. This task is carried out by identifying those leaves of $PrTree$ having an Attractiveness Degree smaller than a certain threshold th''_{Attr}. Let N_k be one of these nodes; removing N_k from P requires that the set $EAHSet_{N_k}$ of AH elements represented in P by N_k must be represented, after the removal of N_k, by the parent N_{P_k} of N_k in $PrTree$. This implies that the Attractiveness Degree $DAttr_{N_{P_k}}$ of N_{P_k} must be updated by summing the contributions of the elements of $EAHSet_{N_k}$. Finally, the i-arc connecting N_{P_k} to N_k is removed.

Computation of the new a-arcs of P. This task consists of the following steps:

- Firstly, a new support list, called *Support History* and denoted by SH, is constructed from H. In particular, for each node h_i of H, a node sh_i of SH is created. Each element sh_i of SH is characterized by: *(i)* its *Identifier* Id_{sh_i}; *(ii)* the *Representative Node Identifier*, denoted by $IdRepres_{sh_i}$, indicating the identifier of the Representative Node of the element ah_i of AH having the same URL as h_i; *(iii)* the *Micro-Session Identifier*, denoted by Id_{MSess}, which identifies the Micro-Session which h_i belongs to. In order to identify this latter, the definitions of Latency Time and Micro-Sessions provided in Section 3 are applied to both the Access Starting Time and the Access Ending Time associated with nodes of H. If, in the thus obtained Support History, there exist two or more consecutive elements having both the same Representative Node Identifier and the same Micro-Session Identifier, they are merged in a unique element.

[3] It is worth pointing out that this value of $DAttr_{N_i}$ is only temporary; it can be modified during the next step of P restructuring.

[4] It is worth observing that, if the association rules corresponding to removed a-arcs would be still valid, there is no information loss since they will be extracted again during one of the next steps of P restructuring.

- The *sequential a-priori* algorithm [4] is applied on the thus obtained SH. In particular, for each element $sh_i \in SH$, an itemset, containing only sh_i, is created in a-priori; for each Micro-Session in SH a sequence is created in a-priori and the Micro-Session Identifier is assumed as the Identifier of the corresponding sequence.
- For each association rule derived by a-priori, an a-arc is added in P, if it does not exist. In more detail, let $sh_i \rightarrow sh_j$ be one of these association rules and let $IdRepres_{sh_i}$ (resp., $IdRepres_{sh_j}$) be the Representative Node Identifier associated with sh_i (resp., sh_j). If an a-arc from the node identified by $IdRepres_{sh_i}$ to the node identified by $IdRepres_{sh_j}$ does not exist in P, it is created and its Aging Coefficient is set to 0; vice versa, if an a-arc between these two nodes already exists in P, its Aging Coefficient is set to 0 and no new arc is created.
- The Aging Coefficient of all a-arcs in P is increased of 1; all a-arcs having an Aging Coefficient greater than a certain threshold th_{Aging} are removed from P.

Restructuring H. Once P has been restructured and, therefore, after all new association rules have been determined, information stored in H has become obsolete. Therefore, all elements must be removed from H in such a way to make it capable of storing information about accesses of u during the next session.

4 Exploiting X-Compass as a Content-Based Recommender System

In this section we illustrate how X-Compass can be exploited as a content-based recommender system. In particular, we describe two different applications, namely: *(i)* the support of a user in identifying the next Web page to visit, taking into account her/his profile and the information stored in the currently accessed Web page; *(ii)* the semantic-driven Web search.

4.1 Supporting the User in Identifying the Next Web Page to Visit

X-Compass is particularly efficient for supporting a user u in her/his choice of the next Web page to visit whenever the following scenario is valid: *(i)* u is visiting a Web page p; *(ii)* links $l_1, \ldots, l_n$, conducting to pages $p_1, \ldots, p_n$ resp., are present in p; *(iii)* u does not know which of pages $p_1, \ldots, p_n$ is the closest to her/his current interests.

In order to support u in her/his choice of the next Web page to visit, the following steps are carried out:

- The Keyword Sets $KSet_{p_1}, \ldots, KSet_{p_n}$, associated with pages p_1, …, p_n, are determined.
- The Similarity Degree $DSim_{p_i N_j}$ is computed, with the support of the function $Max\_Weight\_Matching$, for each pair $(KS_{p_i}, KSet_{N_j})$, where KS_{p_i} is one of the Keyword Sets defined during the previous step and $KSet_{N_j}$ is the Keyword Set associated with a node N_j of P.

- The maximum Similarity Degree $DSim_{p_{max}N_{max}}$, among those computed during the previous step, is determined. Finally, p_{max} is suggested to the user as the next page to visit.

4.2 Semantic-Driven Web Search

Nowadays existing tools for supporting the user in identifying information sources of her/his interest are not capable of providing, so as to say, the “right” answers to a user query but, rather, provide large supersets thereof (consider answers typically returned by Web-search engines). This problem basically relies on syntax-driven retrieval techniques which such tools are based on, whereas semantic-driven search would be needed. As a simple example, consider a user query citing just the term “plant” and assume that this user is interested in vegetables. Then, documents pertaining “plants” in a factory perspective are, most probably, not interesting to her/him: a “semantic-” or “concept-” driven retrieval would be hopefully capable of distinguishing between the two meanings associated with the term “plant” and, therefore, of returning only actually interesting information. Intuitively, any system aiming at carrying out a “semantic-driven” search on the Web must take into account the profile of the user requiring the search. X-Compass is particularly suited in this application context since: *(i)* its ontology consists of the profile of the user associated with it; *(ii)* its knowledge base is stored in XML files and, therefore, the comparison between the Web page information contents and the user profile, as well as the information exchange, are simplified. With the support of X-Compass, it is possible to define the following technique for carrying out the “semantic-driven” search of information sources on the Web:

- The user query is submitted to a traditional Web search engine. Let p_1, ..., p_n be the Web pages returned by it.
- The Keyword Sets $KSet_{p_1}, \ldots, KSet_{p_n}$, corresponding to these pages, are identified.
- The Similarity Degree $DSim_{p_i N_j}$ is computed, with the support of the function $Max\_Weight\_Matching$, for each pair $(KSet_{p_i}, KSet_{N_j})$, where $KSet_{p_i}$ is one of the Keyword Sets defined during the previous step and $KSet_{N_j}$ is the Keyword Set associated with a node N_j of P.
- Pages $p_1, \ldots, p_n$ are filtered on the basis of the computed Similarity Degrees in such a way to select only those semantically interesting for the user. A page p_i is considered semantically interesting for the user if at least one of the Similarity Degrees relative to $KSet_{p_i}$, computed during the previous step, is greater than a certain threshold th'_{Sim}.

It is worth pointing out that X-Compass is much more efficient than other approaches proposed in the literature for carrying out a semantic-driven filtering of the Web pages returned by traditional search engines (such as that described in [14]). The reason of this efficiency can be found in the relative simplicity of the X-Compass approach for constructing and handling the user profile, as well as in the exploitation of XML for storing the X-Compass knowledge base.

5 Exploitation of X-Compass as a Collaborative Filtering Recommender System

In this section we illustrate how, thanks to the choice of storing its knowledge base into XML documents, X-Compass is particularly suited as a collaborative filtering recommender system. In particular, we shall illustrate two X-Compass multi-agent systems, the former devoted to support users for sharing their knowledge and the latter designed for predicting the future interests of a user. Actually, the nature of this last application is quite different from the nature of the other three mentioned previously since it has been designed for operating on the "server" side rather than on the "client" side. Nevertheless, we argue that it is particularly interesting and could be the starting point for further future applications.

5.1 Supporting Users for Sharing Their Knowledge

An X-Compass multi-agent system for supporting knowledge sharing consists of n X-Compass agents $Ag_{xc}(u_1), \ldots, Ag_{xc}(u_n)$, each associated with a user. Let $N_{i_1}, \ldots N_{i_{card(i)}}$ be the nodes of the User Profile $P(u_i)$ relative to $Ag_{xc}(u_i)$[5]. Let $KSet_{N_{i_1}}$, …, $KSet_{N_{i_{card(i)}}}$ be the Keyword Sets associated with nodes $N_{i_1}, \ldots, N_{i_{card(i)}}$. Cooperation among X-Compass agents for supporting knowledge sharing is based on a protocol consisting of the following steps:

– Function $Max\_Weight\_Matching$ is applied on each pair of Keyword Sets $(KSet_{N_{i_s}}, KSet_{N_{j_t}})$ such that N_{i_s} and N_{j_t} are nodes belonging to the User Profiles $P(u_i)$ and $P(u_j)$ with $i \neq j$; in this way, the Similarity Degree $DSim_{st}$ between the interests associated with nodes N_{i_s} and N_{j_t} is derived.
– If there exist two nodes N_{i_s} of $P(u_i)$ and N_{j_t} of $P(u_j)$ such that $DSim_{st}$ is greater than a certain threshold th''_{Sim}, the multi-agent system concludes that u_i and u_j have a common interest; therefore u_i (resp., u_j) can provide u_j (resp., u_i) with those pages of $AH(u_i)$ (resp., $AH(u_j)$) represented by N_{i_s} (resp., N_{j_t}) in $P(u_i)$ (resp., $P(u_j)$).

5.2 Predicting Future Interests of a User

An X-Compass multi-agent system capable of predicting the future interests of a user can be easily defined by taking into account both the semantics and the update policy of the User Profile in X-Compass. Recall that nodes of a User Profile represent interests; when the Attractiveness Degree of a node is quite high, it is split and, after such a task, new nodes, and therefore new interests, appear in the User Profile (see Section 3.2).

An X-Compass multi-agent system for predicting the future interests of a user is composed by n X-Compass agents $Ag_{xc}(u_1), \ldots, Ag_{xc}(u_n)$, each associated with a user. Let u_i and u_j be two users and assume that two nodes N_{i_s} of $P(u_i)$ and N_{j_t} of $P(u_j)$ exist, each characterized by a high Attractiveness Degree and, therefore, next to be split. Assume that the Similarity Degree between

[5] Here, function $card(i)$ returns the cardinality of the set $NS(u_i)$ of nodes of $P(u_i)$.

N_{i_s} and N_{j_t}, computed by applying the function $Max\_Weight\_Matching$ in a way analogous to that we have seen in the previous sections, is high. Assume, moreover, that N_{i_s} is split and that new nodes, and therefore new interests, appear in $P(u_i)$. Since the Similarity Degree between N_{i_s} ed N_{j_t} is high, it is possible to presume that when N_{j_t} will be split (this event will happen soon since the Attractiveness Degree of N_{j_t} is high) the interests appearing in $P(u_j)$ will be analogous to those just appeared in $P(u_i)$. Therefore, from the analysis of the new interests of u_i, it is possible to predict the next interests of u_j.

The capability of predicting future interests of a user can be exploited in a large variety of application contexts; as an example, an e-commerce site could benefit of such a capability for predicting future interests of its customers and, consequently, for suitably personalizing its offers; in this way, it would be capable of anticipating its competitors and, consequently, of acquiring a strong competitive advantage. But this is only one among many possible applications; actually, any system someway exploiting the knowledge of the profile of its users could acquire a competitive advantage from this prediction capability.

6 Conclusions

In this paper we have presented X-Compass, an XML agent for supporting a user during her/his navigation on the Web. In particular, we have seen that: *(i)* X-Compass is capable of constructing and managing the profile of the user associated with it by examining her/his past behaviour; *(ii)* thanks to its features, X-Compass can be exploited as both a content-based and a collaborative filtering recommender system; *(iii)* since X-Compass is based on XML, it is particularly light and, at the same time, well suited for supporting data exchange and, therefore, agent cooperation. Presently we are working for implementing the X-Compass prototype on different hardware and software platforms such as workstations, personal computers and palmtops; the capability of operating on a large variety of platforms, often characterized by reduced resources, is gained by exploiting XML, instead of a classical database, for storing the X-Compass knowledge base. The prototype availability shall allow us to carry out a large variety of tests some of which will be devoted to determine the (presumably dynamic) values of the exploited thresholds.

As far as our future work is concerned, we argue that X-Compass could have further interesting applications in those contexts where the knowledge of a user profile plays a key role; among these we plan to soon investigate e-commerce and telecommunications. As for this last context, it appears particularly promising to exploit X-Compass for managing data transmission when a bandwidth restriction has occurred; indeed, we argue that X-Compass could exploit the knowledge stored in the user profile for determining those parts of the information to transfer that the user considers particularly important (which, therefore, should be transferred in any case) as well as that information considered less attractive by her/him (which, consequently, might not be transmitted or, alternatively, might be transmitted later).

References

1. *Communications of the ACM, Special Issue on Recommender Systems*, volume 40. ACM, 1997.
2. G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *IEEE Computer*, 34(2):74–82, 2001.
3. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of International Conference on Management of Data (SIGMOD'93)*, pages 207–216, Washington, D.C., USA, 1993. ACM Press.
4. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of the International Conference on Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society.
5. A.G. Buchner and M.D. Mulvenna. Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Record*, 27(4):54–61, 1998.
6. L.D. Catledge and J.E. Pitkow. Characterizing browsing strategies in the world-wide web. *Computer Networks and ISDN Systems*, 27(6):1065–1073, 1995.
7. Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Computing Surveys*, 18:23–38, 1986.
8. R.J. Glushko, J.M. Tenenbaum, and B. Meltzer. An XML framework for agent-based e-commerce. *Communications of the ACM*, 42(3):106–114, 1999.
9. N.R. Jennings and M.J. Wooldrige (eds.). *Agent Technology: Foundations, Applications, and Markets*. Springer-Verlag, 2002.
10. T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proc. of the International Conference on Machine Learning (ICML'97)*, pages 143–151, Nashville, USA, 1997. Morgan Kauffman.
11. L. Kaufman and P.J. Rousseeuw. *Findings Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, New York, 1990.
12. A.G. Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
13. B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on Web usage mining. *Communications of the ACM*, 43(8):142–151, 2000.
14. L. Palopoli, D. Rosaci, G. Terracina, and D. Ursino. Modeling web-search scenarios exploiting user and source profiles. *AI Communications*, 14(4):215–230, 2002.
15. G.A. Papadopoulos. Models and technologies for the coordination of Internet agents: A survey. In *Coordination of Internet Agents: Models, Technologies, and Applications*, pages 25–56. Springer-Verlag, 2001.
16. J.S. Park and R.S. Sandhu. Secure cookies on the web. *IEEE Internet Computing*, 4(4):36–44, 2000.
17. M. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408, 1999.
18. G. Salton and M.G. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.
19. U. Shardanand and P. Maes. Social information filtering:algorithms for automating 'word of mouth'. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 194–201, New York, 1995. ACM Press.
20. J.M. Tenenbaum, T.S. Chowdhry, and K. Hughes. Eco system: An internet commerce architecture. *IEEE Computer Journal*, 30(5):48–55, 1997.

A Parallel Spatial Join Processing for Distributed Spatial Databases

Myoung-Soo Kang[1], Seung-Kyu Ko[1], Kyun Koh[2], and Yoon-Chul Choy[1]

[1] Department of Computer Science, Yonsei University, 134, Seoul, 120-749, Korea
{soo, pitta, ycchoy}@rainbow.yonsei.ac.kr

[2] Department of Computer and Information Engineering, Chongju University, Cheongju, Chungbuk, 360-764, Korea
kyunkoh@chongju.ac.kr

Abstract. In recent years, there have been needs of accessing spatial data from distributed and preexisting spatial database systems interconnected through a network. In a distributed environment, spatial joins for two spatial relations residing at geographically separated sites are expensive in terms of computation and transmission cost because of the large size and complexity of spatial data. Previous distributed algorithm based on the spatial semijoin has accomplished performance improvements by eliminating objects before transmission to reduce both transmission and local processing costs. But with a widespread of a high bandwidth data transmission, the parallelism through data redistribution may improve the performance of spatial joins in spite of additional transmission costs. Hence, we propose a parallel spatial join processing for distributed spatial databases. We apply the task distribution method minimizing the data transmission and the solution for task distribution using a graph partitioning method. In experiments, we showed that the proposed method provides useful reductions in the cost of evaluating a join.

1 Introduction

The advent of the Internet and the World Wide Web has given rise to a wide variety of distributed applications, one being geographic information systems (GIS). In recent years, there have been needs of accessing spatial data from distributed and preexisting spatial database systems interconnected through a network.

A distributed GIS is a group of sites, which are geographically separated but connected with a data communication network. Each site is an autonomous GIS, which has processing power and geospatial data [1]. For a user, a distributed GIS may provide transparent access to the data stored at both the local and remote sites. A distributed GIS may well support cooperative projects among different organizations. A distributed GIS has potential in reliability, parallelism, efficiency, and resource sharing.

The cost of spatial join processing could be very high due to the large sizes and complexity of spatial objects involved. In a distributed environment, spatial joins for two spatial relations residing at geographically separated sites are expensive in terms of

transmission cost as well as computation cost. Although there exists a variety of algorithms for the join processing for single-site databases, not much research has been done to improve the performance of joins for distributed databases. In this paper, we focus on the design of distributed spatial join algorithms.
Unlike conventional distributed databases where the transmission cost is dominant, both the transmission cost and the processing cost may be important for distributed spatial databases. Previous distributed spatial join algorithm using semijoin is based on the concept of spatial semijoin, so that a spatial semijoin eliminates objects before transmission to reduce both transmission and local processing costs [2,3].
But with a high bandwidth data transmission, the parallelism through data redistribution can improve the performance of spatial joins in spite of additional transmission costs [4,5]. So we propose a parallel spatial join strategy, which have proven successful for parallel systems, for distributed spatial databases with certain modifications. As it does in parallel systems, in distributed systems data partitioning parallelism can be explored for join operation by partitioning join relations into buckets and redistributing tasks (pairs of buckets) between two sites and applying the join operator on tasks in parallel at each site. But, unlike parallel systems, the distributed systems have the data to be distributed between two sites before join processing starts and the initial data distribution between two sites to be not even. In this paper we concentrate on the task decomposition method with particular attention to examining how it can minimize the size of data, which are redistributed, and the number of duplicate objects produced by the data partitioning. The basic idea is that, firstly, because transferring relations that are small reduce the size of transmitted data, the overall execution time also can be reduced. Secondly, we transform task decomposition problem into graph partitioning problem. The optimal assignment-which corresponds to the minimum weight cutset-can be found very efficiently using any one of available network flow algorithms.
In this paper, we assume that the processing capability for each site is the same and there are no other processes other than spatial join.
The rest of this paper is organized as follows. In Section 2 we survey related works on distributed spatial join. The parallel approach for distributed spatial join is given in Section 3, together with data-partitioning technique. Section 4 proposes the task assignment method based on the graph model. Section 5 compares the experimental results for performance time with existing methods and Section 6 describes conclusion and further works.

2 Related Works

2.1 Spatial Joins and Filter-and-Refinement Algorithm

The Spatial join operation combines two or more sets of spatial objects according to spatial predicates. Spatial predicates used in spatial join algorithm are “intersection”, “containment”, “distance within” and etc. In this paper, we are primarily interested in spatial intersection joins, which are considered to be the most important type of spatial join. In order to reduce its complexity, the spatial join is processed in two steps. In the *filter* step, a spatial join is performed using conservative approximations of the spatial objects. The filter step produces a candidate set that composing the super set of correct results of the spatial join. In the *refinement step*, the candidate objects are tested using their exact geometry whether they really satisfy the join predicate. The most commonly used approximations are Minimum Bounding Rectangles (MBRs) [6,7].

In the following, we consider two inputs of the join, $R = \{r_1, r_2, \ldots, r_n\}$ *and* $S = \{s_1, s_2, \ldots, s_m\}$ of spatial objects. We assume that the system has a unique identifier for each object. Let $Id(r_i)$ and $Mbr(r_i)$ denote the identifier and MBR of a spatial object, r_i, respectively. *Id* is a function that assigns a unique identifier to each spatial object in a database and *Mbr* is a function that computes the MBR of a spatial object. MBR-spatial join compute every pair $(Id(r_i), Id(s_j))$ where $Mbr(r_i) \cap Mbr(s_j) \neq \varnothing$. Finally, spatial joins compute all pairs $(Id(r_i), Id(s_j))$ with $r_i \cap s_j \neq \varnothing$ using the results of filter step.

The filter-and-refinement approach can improve the execution time with respect to both, CPU- and I/O time [6,7].

2.2 Previous Distributed Spatial Join Algorithms

While spatial database research to date has largely focused single-site environment, which has uniprocessor or parallel processors in one system, not much research has been done to improve the performance of spatial joins for distributed databases. To our knowledge, there is no reported work on distributed spatial join processing except for the semijoin-based approach [2,3].

Unlike single-site environment, in distributed spatial databases, the transmission cost is important because of relatively slow networks. So there needs methods that can improve the performance of spatial join by minimizing the size of data transmission.

For the sake of concreteness, the followings are assumed for distributed spatial join: let *R* and *S* denote the two inputs to the join. *R* resides at R_{site} and *S* resides at S_{site}.

The ship whole strategy is the basic approach for distributed spatial join. It transmits *S* to R_{site}. Then spatial join is performed between *R* and *S* on R_{site}. It is relatively simple method. But it has disadvantages that both the transmission cost and the local processing cost are very high.

Distributed spatial join method based on semijoin transmits only approximations (MBRs) of *S* objects to R_{site}. Then MBR-spatial join is performed at R_{site} between MBRs of *R* objects and MBRs of *S* objects to be transmitted from S_{site}. We can obtain candidate objects from this filter step. Afterwards, geometric information of *R* objects belong to the candidate set are sent to S_{site}. In the refinement step, the candidate objects are tested using their exact geometry whether they really satisfy the join predicate at S_{site}. Distributed spatial join based on semijoin can be defined as follows.

(1) $S' = \{Mbr(s) \mid s \in S\}$
(2) Transmit S' information from S_{site} to R_{site}
(3) $R' = \{r \in R \mid \exists s \in S'$ such that $Mbr(r) \cap s\}$

(4) Transmit *R'* to S_{site}
(5) Perform join refinement step on *R*' and *S*

This method has accomplished performance improvements by eliminating objects before transmission to reduce both transmission and local processing costs.
Both the ship-whole strategy and semijoin-based strategy do not explore parallelism, one of the potentialities of distributed systems. This paper studies the use of data partitioning parallelism, which has proven successful for parallel systems, in order to improve the response time of distributed spatial join queries.

3 Parallel Join Strategy in Distributed Spatial Database

One of the main advantages of distributed database is parallelism of work. Technically, a query can be performed at several sites simultaneously, potentially making the total time required for the query less than when the query were performed on a single site in a sequential manner. With a centralized database, a query is performed on a single machine on each of the relations that are specified. In a distributed environment, the query can be sent to machines where the data resides and the query can be done in parallel. The results will eventually be collected at the querying site. If a query takes 10 units of time on a single machine, an optimally parallel query on two machines take 5 units of time because the work is divided between the two machines. Since communication is required to send the query to the nodes and to send the result back, some communication time must be taken into account. The join query results in the extra communication cost to execute the join query in parallel because two join operand relations need to be at the same node and it is required to redistribute join relations among nodes. This network load could be detrimental to performance.
In this section, we propose parallel spatial join strategy for distributed database. There are several parallel join algorithms for spatial data. But we concentrate on the difference of condition between distributed and parallel environment and propose a method to improve the performance of parallel strategy in distributed environment.
We first review existing parallel join algorithms. In particular, we pay attention to join algorithms that employ data partitioning parallelism and the filter-and-refinement for spatial join processing, as our distributed spatial join strategies are based on them.

3.1 Parallel Spatial Join Processing: A Brief Review

The parallel processing of spatial joins has been increasingly studied. Brinkhoff proposed a spatial join algorithm using an R*-tree in the parallel system with shared-virtual memory [10]. Hoel presented a parallel processing of spatial joins using PMR Quad-tree that is the multi-assignment spatial index to regularly decompose the data space [11]. The algorithm is restricted to operate on a special-purpose platform (SIMD). Zhou also introduced a framework for parallel processing of spatial joins using the multi-assignment index [8,9]. Kim has designed two kinds of parallel spatial

join algorithms based on the grid files: a parallel spatial join using a multi-assignment grid file and a parallel spatial join using a single-assignment grid file [12,13].
Their researches except Zhou rely on preexisting indices. In this paper, we do not assume any spatial index. It enables to handle the case where the join operand data are the results of some previous operations, or where data can be bulk loaded into a site from a outside (e.g., a database server or a file system). So our distributed spatial join is based on the data partitioning method for parallel spatial join processing suggested by Zhou.

3.2 Data Partitioning Parallelism for Distributed Spatial Join

Data partitioning method enables us to explore CPU parallelism in data parallel environment. For data partitioning parallelism we partition the input data and apply the operator on part of data in parallel. Because the data are distributed between two sites before join processing starts, we have the data redistribution phase such that each site can execute their tasks, subqueries in parallel. In the data redistribution phase, the join operand data are partitioned into buckets, and bucket pairs are grouped into tasks.
The architecture of parallel join execution is shown in following Figure 1.

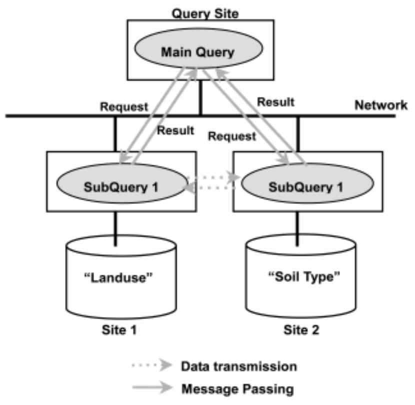


Fig. 1. Parallel Processing in Distributed Spatial Database

The main query is issued on the query site. The main query sends query parameters to the two sites that the join relation resides at and waits for results from each site after sending a "query execution" command. After the data redistribution stage, subqueries are executed in parallel at each site. The relevant sites are activated simultaneously. When all the results from the subquery sites are collected, it merges the results to give overall answers. It is assumed that processing capability for each site is the same and there are no other processes other than the spatial join.

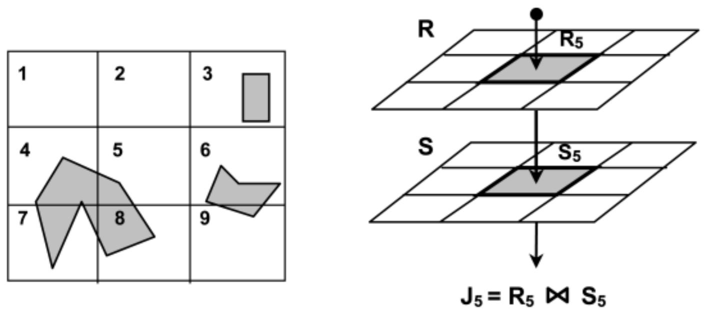
3.2.1 Spatial Data Partitioning

Figure 2 shows that the spatial extents of layers *R* and *S* are divided into *9* equal-sized cells for concise description. The number of cell division can be extended to any number. A spatial object that is entirely contained within a cell is assigned to the corresponding bucket. Spatial objects that overlap multiple cells are replicated in all the buckets that correspond to the cells that the object overlaps. Many researches use this data partitioning method to map all objects into a bucket for a cell [8,9,14,15,16].

As all the spatial objects overlapping with a cell is mapped to the corresponding bucket, *R* and *S* is divided into buckets, $R_1, R_2, \ldots R_n$ and $S_1, S_2, \ldots, S_n$, respectively such that for all $1 \leq i, j \leq n$, $R_i \cap R_j \neq \varnothing$ and $S_i \cap S_j \neq \varnothing$. There is overlap between buckets because information for spatial objects that overlap multiple cells are stored redundantly in the buckets of corresponding cells.

Let $|R|$ represent the cardinality of the input *R*, and $|S|$ represent the cardinality of the input *S*. And let α and β be the rate of repetition for an object of *R* and *S*. The size of buckets R_i and S_i can be approximated to $(1+\alpha)|R|/n$ and $(1+\beta)|S|/n$, respectively.

As spatial join, J_i, between the corresponding buckets, R_i and S_i is defined as a *task*, the result of spatial join between *R* and *S* can be obtained by the sum of the result of tasks, $\{J_1, J_2, \cdots, J_n\}$. In other words, it can be defined as $R \infty S = \bigcup_{i=1}^{n} (R_i \infty S_i)$. The subquery J_5 in Figure 2 means the join task between corresponding buckets R_5 and S_5.



(a) Spatial division by 3*3 cells (b) subquery J_5

Fig. 2. Spatial Data Partitioning

3.2.2 Overall Framework

The following describes the entire process for the proposed distributed spatial join. We follow the general approach of previous parallel spatial join processing method in [8,9]. The join is processed in two steps: filter and refinement. In each step, task creation and distribution is executed in parallel at two sites. In each step, task assignment is included to minimize both the communication time and duplicate objects. After the execution of each step, duplicate results are eliminated. Figure 3 shows the data redistribution phase after spatial data partitioning.

[1] Filter step

① Filtering task decomposition

The approximation data are partitioned in parallel into buckets, and bucket pairs are grouped into tasks. We can estimate the workload of the filtering task by the number of spatial objects belongs to the cell, because the size of the approximation of the spatial object, MBR, is the same.

② Approximation data redistribution for filtering step

Approximations of spatial objects are redistributed between two sites, such that each site can execute filtering task independently. The approximation data is composed of a set of <id of object, MBR>.

③ Parallel filtering in two sites

Each filter tasks assigned to two sites are processed in parallel.

[2] Refinement step

① Refinement task decomposition

Task is divided so that equation (3) can be satisfied on candidate objects obtained from the filter step. The number of edges of spatial object is used to approximate the workload of the refinement task.

② Full object redistribution for refinement step

Spatial objects are redistributed between the two sites, such that each site can execute refinement task independently.

③ Parallel refinement in two sites

Each site performs its refinement task using polygon intersection algorithm.

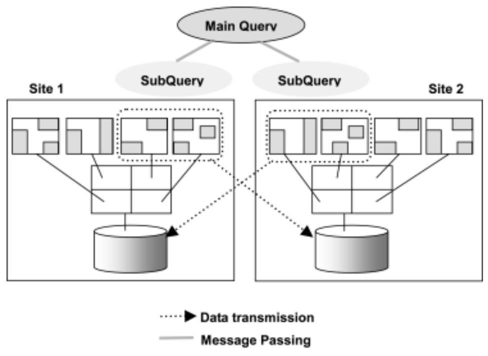


Fig. 3. Data Redistribution for Parallel Processing

4 Task Distribution Methods

After spatial data partitioning, a group of tasks is appropriately distributed to two sites and executed in parallel. In this chapter, we deal with the task assignment problem. The objective is to find assignments with the minimum total sum of execution and communication costs.

We present a task distribution method minimizing the data transmission and the solution for task distribution using a graph partitioning method. Our issues include estimating the workload of distributed spatial joins.

4.1 Cost Model

The cost for a distributed join can be divided into communication cost and processing costs.
We use an assignment function $X\colon J_i \rightarrow S_{X(i)}$ to represent the assignment of the *n* tasks to the sites. The cost of the assignment *X* is the sum of the total execution and communication costs, *i.e.*,

$$Cost(X) = CExec(X) + CComm(X) \tag{1}$$
$$= \Sigma\, CExec(R_{site}) + \Sigma\, CExec(S_{site}) + \Sigma\, Comm(R_{site,}\ S_{site})$$

Then, the task assignment problem is to find an assignment X_0 with the minimum cost, *i.e.* $Cost(X_0) = min_X Cost(X)$.

The communication cost of information transfer per unit between two sites is known to be 'K_c' and is assumed to be constant. Without loss of generality, the value of 'Kc' can be assumed to equal to 1. That is, the cost of communication is directly proportional to the number of bytes transmitted. If we define the average size of an object transmitted, the communication cost can be computed in terms of the number of objects(N_{Obj}), as follows:

$$CComm = K_c * N_{Obj} \tag{2}$$

Depending on $X(J_i)$, N_{Obj} is different in size. If $X(J_i)$ is R_{site}, N_{Obj} is $|S_i|$. And if $X(J_i)$ is S_{site}, N_{Obj} is $|R_i|$. We can reduce the communication cost by assigning the smaller of the size of $|S_i|$ and $|R_i|$ as $X(J_i)$.
The probability of a rectangle $r = (r_1, \ldots, r_n)$ intersecting another rectangle *s* that has the sides $(s_1, \ldots, s_n)$ is estimated by the following formula:

$$P(\cap(r,s)) = \prod_{k=1}^{d} (r_k + s_k) \tag{3}$$

When joining two relations, the expected number of pairs is approximated as:

$$|R| * |S| * \prod_{k=1}^{d} (r_k + s_k) \tag{4}$$

Where $|R|$ and $|S|$ is the cardinality of the *R* and *S*, respectively.

The spatial join method uses two join algorithms in two phase of join processing: MBR intersection join algorithm for obtaining candidate objects in the filter step and polygon intersection join algorithm for obtaining the actual result in the refinement step. Equation (4) can be approximated by $|R|*|S|$ in the filter step because MBR has *d* equal to a constant '4', while it being $|R_D|*|S_D|$ in the refinement step. $|R_D|$ and $|S_D|$

means the dimension of polygons, and can be approximated by the number of vertexes. Table 1 shows the execution cost and the communication cost produced by cost model described above.

4.2 Graph-Based Approach

Graph theory is a branch of mathematics that has found numerous applications in many different fields. It deals with entities (nodes), and the connections between entities (edges). Graph theoretic techniques have been successful in modeling many problems of assignment and partitioning in distributed systems, since the notions of 'node' and 'edge' from graph theory are very similar to the concepts of 'modules' and 'communication' in distributed programs. In this chapter, we propose the graph model for spatial join tasks in distributed environments and apply the graph-partitioning solution for the problem of assigning tasks over sites. The problem is thus one of splitting a task set into two parts, such that the sum of all the execution costs of all tasks and all the overlap costs of all tasks is minimum.

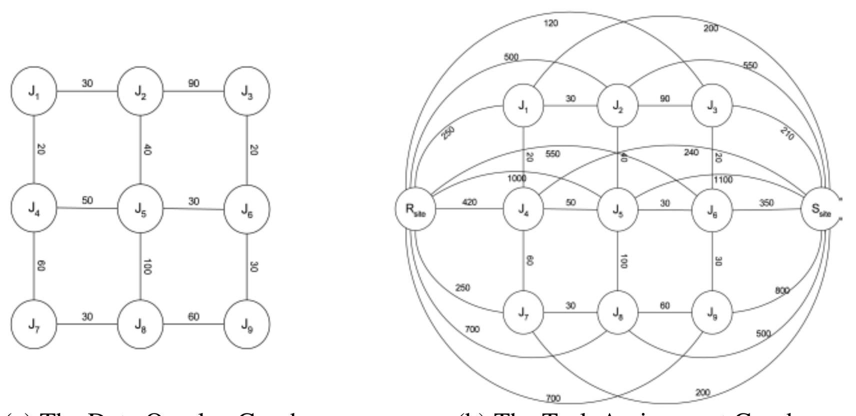
Herald Stone pioneered the application of network flow algorithm to task assignment problem in distributed computer systems. He showed how an assignment problem could be transformed into a network flow problem such that there is a one-to-one correspondence between assignments and cutsets [17]. For the case of two processor problems, the optimal assignment-which corresponds to the minimum weight cutest-can be found very efficiently using any one of available network flow algorithms [18].

We define a data overlap graph to represent the interaction among tasks. It has nodes corresponding to each task and has an edge between two nodes if and only if the corresponding tasks are adjacent. For each pair J_i and J_j belonging to the data overlap graph, we define a weight $D_{ij}(=D_{ji})$ to depict the amount of duplicate objects between J_i and J_j resulting from the data partitioning.

We proceed to construct a two-terminal network flow graph called a task assignment graph. This graph contains the data overlap graph of Figure 4(a) and has two additional nodes and $2n$ additional edges (recall that n is the number of tasks in a join query). Two additional nodes are labeled R_{site} and S_{site} and represent the two sites. The additional edges all extend from these nodes to the nodes representing tasks. We call these *execution edges*; the original edges of the data overlap graph are called *overlap edges*.

Table 1. The Execution Cost and the Communication Cost

	Execution cost	Communication cost	
		R_{site}	S_{site}
J_1	150	50	100
J_2	350	200	150
J_3	110	100	10
J_4	220	20	200
J_5	700	400	300
J_6	300	50	250
J_7	150	50	100
J_8	400	100	300
J_9	500	300	200



(a) The Data Overlap Graph (b) The Task Assignment Graph

Fig. 4. Graph Model

The overlap edges of the assignment graph bear weights that are the same as the weights on the corresponding edges in the data overlap graph of Figure 4(a). These represent the data overlap costs. The execution costs are placed on the execution edges as follows. The edge connecting node J_i to node R_{site} is labeled with the cost of executing task J_i on site S_{site} and vice-versa. For example in Figure 4(b), the edge joining node J_1 and R_{site} is labeled with the cost of executing task on processor S_{site}. This reversal of labeling is intentional.

Task assignment graphs drawn up in the fashion described above have the following important property: a cut that disconnects R_{site} and S_{site} corresponds to an assignment of modules to processors and vice versa. After removing the edges of the cut from the graph, the nodes that are reachable from R_{site} (S_{site}) are considered to be assigned to site R_{site} (S_{site}). There is in fact a one-to-one correspondence between cuts and assignments.

If a graph G has two distinguished nodes s and t an if a cutset breaks G into two components G_1 and G_2 such that s is contained in G_1 and t in G_2, then the cutset is called an *s-t cut*. A *minimum weight s-t cut* or *mincut* in a graph is an s-t cut with the minimum weight in the graph. In order to find the optimal or minimum assignment, we need to find the mincut in the assignment graph. This is done by applying a maximum flow algorithm with R_{site} as s and S_{site} as t.

The weight of a cut in the assignment graph is equal to the total cost of the corresponding module assignment [Stone77]. This theorem indicates that an optimal assignment can be found by running a network flow algorithm on the assignment graph. A network flow algorithm applied to the graph shown in Figure 4(b) produces the minimum weight cutset. Note that the weight of the cut is equal to the total cost of the corresponding assignment.

5 Experimental Results

5.1 Environment

For experiments, two Personal Computers were used, each having 1.2GHz Pentium III processor, 256MB memory and 20GB disk. This system is designed and implemented based on the Common Object Request Broker Architecture (CORBA). The code is developed using Visual C++ 6.0 and Visibroker for C++ 4.0 [19] under Windows 2000.

We conducted experiments using the Sequoia 2000 benchmark data set [20], which are publicly available and have been used by several published spatial join analysis. It consists of two sets of polygons: the "land use" polygons and the "island" polygons. We denote the "land use" polygons as R, and the "island" polygons as S hereafter. The dataset contains 79,607 polygons (|R|=58,586, |S|=21,021) with 3.65 million vertices in total. The data is in a form of a typical spatial data consisted of at least 3 to at most 8400 vertexes.

5.2 Results

Following the distributed join strategies described in Section 3.1, we studied the following three algorithms.

- Algorithm J-W. This is the ship whole algorithm that transmits R to S_{site}, and performs the join directly.
- Algorithm J-S. This algorithm uses a semi-join. MBR of each object of S is used as its approximation. The MBRs are transmitted to R_{site} to reduce R before R is sent to S_{site} for the final join.
- Algorithm J-P. This is the parallel processing algorithm through data redistribution techniques.

From Figure 5 and Figure 6, we see that the parallel processing strategy is superior to others. Recall that the parallel processing strategy requires the additional data transmission in order to redistribute data between sites.

Our results showed that the processing time saved by parallel join is predominant to these communication overheads incurred.

We examine the effects of the two parameters: network bandwidth and join selectivity. Figure 5 shows the effect of the network bandwidth on the three algorithms. As the network bandwidth decreases, the transmission cost of the data that are passed around also increases, and consequently, the performance of the J-W and the J-S rapidly degrade. The J-P shows the performance improvement of maximum 33% than J-S.

Join Selectivity is the ratio of the cardinality of the output relation to the product of the cardinalities of the input relations. As the join selectivity increases, the sizes of the candidate sets that are passed around also increases, thereby, causing join processing to perform poorly. We examine the effect of join selectivity on the two algorithms.

To obtain the desired effect, we again joined the two layers, but arbitrarily dropped half of the candidate set that are produced during the execution of the filter step. This process of reducing the cardinality of the candidate set by half has the effect of approximately halving the cardinality of the final result set. Figure 6 plots the effect of the join selectivity on the two algorithms for two join selectivity's: 0.4e-6 and 02e-6. As the join selectivity decreases, the size of the candidate sets that are passed around also decreases, the performance of the J-S rapidly upgrades. Decreasing the join selectivity has little affect on the performance of J-R because the effect of CPU parallelism decreases.

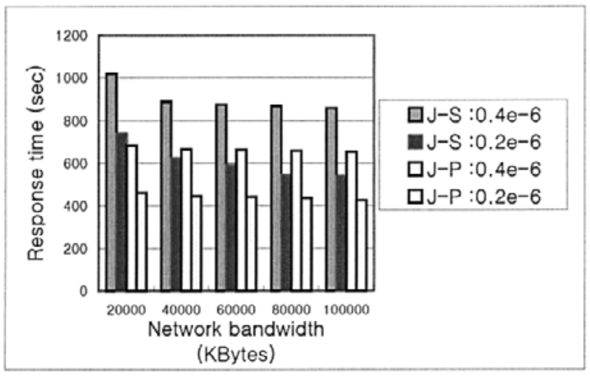


Fig. 5. Comparisons of distributed spatial join algorithms

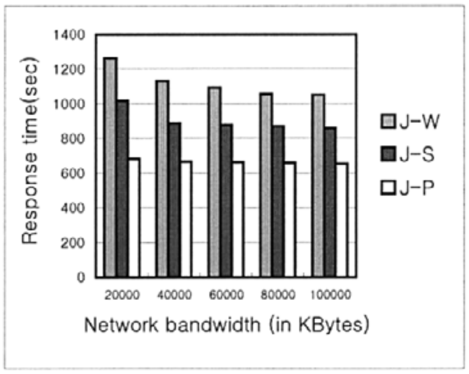


Fig. 6. Effect of Join Selectivity (Join Selectivity = 0.4e-6)

6 Conclusions

The paper proposed an effective spatial join method for layers stored in separate sites connected by network. This is an important issue because of different organizations of GIS need to be linked with one another or other information systems.

The concept of the paper was based on the fact that when query in distributed system is processed in parallel, the response time is faster. To process queries in parallel, data stored in different sites need to be redistributed. But, even with communication overhead due to redistribution, faster response time can be obtained through parallel processing of query.

Unlike parallel systems, the distributed systems have the data to be distributed between two sites before join processing starts and the initial data distribution between two sites to be not even. In this paper we concentrate on the task decomposition method with particular attention to examining how it can minimize the size of data, which are redistributed, and the number of duplicate objects.

To appropriately redistribute spatial data to each site, we use the data-partitioning techniques used by most parallel join algorithms in relational databases. By applying the proposed distributed spatial join technique, load for the join process can be distributed to related sites and the performance can be improved.

We investigate two factors: network bandwidth and join selectivity. As the network bandwidth decreases, the transmission cost of the data that are passed around also increases, and consequently, the performance of the J-W and the J-S rapidly degrade. The J-P shows the performance improvement of maximum 33% than J-S. And as the join selectivity decreases, the size of the candidate sets that are passed around also decreases, the performance of the J-S rapidly upgrades. Decreasing the join selectivity has little affect on the performance of J-R because the effect of CPU parallelism decreases.

In our future work, we plan to consider distributed spatial join strategy for the heterogeneous system where nodes are not identical, and for each node, the processing ability is different and there is a variety of workloads, involving not only spatial joins but other kinds of queries also. We shall report such a study in a future paper.

References

1. Wang, F.: A Distributed Geographic Information System on the Common Object Request Broker Architecture. GeoInformatica, Vol. 4, No.1, (2000) 89-115
2. Abel, D.J., Ooi, B.C., Tan, K.L., Power, R., Yu, and J.X.: Spatial Join Strategies in Distributed Spatial DBMS. Proceedings of 4th International Symposium on Large Spatial Databases (SSD), Portland, USA, (1995) 348-367
3. Tan, K.L., Ooi, B.C., and Abel, D.J.: Exploiting Spatial Indexes for Semijoin-Based Join Processing in Distributed Spatial Databases. IEEE Transactions on Knowledge and Data Engineering, Vol.12, No.6, (2000) 920-936
4. Park, C. and Seret, D.: Parallel Join Processing in Broadcast Lacal Area Network, Proceedings of the IEEE Region 10 Conference (TENCON), Seoul, (1987)
5. Wond, E. and Katz, R.H., Distributing a Database for Parallelism, Proceedings of the ACM-SIGMOD International conference on Management of Data, (1983)
6. Brinkhoff, T., Kriegel, H.P., and Seeger, B.: Efficient Processing of Spatial Joins Using R-trees. Proceedings of ACM SIGMOD Conf., (1993)

7. Brinkhoff, T. et al.: Multi-Step Processing of Spatial Joins, Proceedings of the ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, (1994) 197-208
8. Zhou, X., et al.: Data Partitioning for Parallel Spatial Join Processing, Proceedings of 5th International Symposium on Large Spatial Databases (SSD), (1997) 178-196
9. Zhou, X., Abel, D.J., and Truffet, D.: Data Partitioning for Parallel Spatial Join Processing. GeoInformatica (1998) 175-204
10. T. Brinkhoff, H.P. Kriegel and B. Seeger: Parallel Processing of Spatial Joins Using R-trees, International Conference on Data Engineering, New Orleans, (1996) 258-265
11. E. G. Hoel and H. Samet: Data-Parallel Primitives for Spatial Operations using PM-Quadtrees, Proceedings of Computer Architectures for Machine Perception '95, Como, Italy, September (1995) 266-273
12. Jin-Deog Kim, Bong-Hee Hong: Parallel Spatial Joins using Grid Files, Proceedings of the Seventh International Conference on Parallel and Distributed Systems (ICPADS'00)
13. Jin-Deog Kim, Bong-Hee Hong: Parallel Spatial Join Algorithms using Grid Files, Proceedings of the International Symposium on Database Applications in Non-Traditional Environments (DANTE'99), (1999) 226-234
14. Shekhar,S.,et al.: Declustering and Load-Balancing Methods for Parallelizing GIS, IEEE Transactions on Knowledge and Data Engineering, Volume 10, No.4, (1998) 632-655
15. Patel, J. M. and Dewitt,D. J.: Partition Based Spatial-Merge Join, Proceedings of ACM SIGMOD International conference on Management of Data (1996)
16. Patel, J., et al.: Building a scalable geo-Spatial DBMS: Technology, implementation, and Evaluation, Proceedings of ACM SIGMOD International conference on Management of Data, SIGMOD '97, (1997)
17. Stone, H.S.: Multiprocessor scheduling with the aid of network flow algorithms, IEEE Transactions on Software Engineering, vol. SE-3, (1977) 85-93
18. Ford, L.R., and Fulkerson, D.R.: Flows in Networks, Princeton, NJ: Princeton Univ. Press, (1962)
19. Borland Corporation, Visibroker Home page. http://www.borland.com/bes/visibroker/.
20. Sequoia 2000 FTP server Home page. http://s2k-ftp.cs.berkeley.edu:8000/.

Query Construction through Meaningful Suggestions of Terms

E. Kapetanios and P. Groenewoud

Dept. of Computer Science,
ETH-Zentrum, Postfach 8092,
ETHZ, Switzerland
kapetanios@inf.ethz.ch
http://www.inf.ethz.ch/personal/kapetani

Abstract. Query formulation by using database specific query languages such as SQL or OQL turns out to be cumbersome or even impossible when end-users need to pose queries to large database schemes. This is due to the difficulties which arise out of the wrong or impossible interpretations of storage models and the lack of mechanisms to embed application domain semantics within query languages. Visual query languages (VQLs) and natural language (NL) based query interfaces in query answering systems alleviate, in some cases, the task of constructing a query. VQLs, however, are bound to visual formalisms which need further interpretation and still lack the use of semantics other than those provided by well-known conceptual models (EER, OMT, etc.). NL based approaches, on the other side, presuppose a considerable knowledge of the vocabulary terms to be used by the end-user for a particular application domain and, furthermore, they do not exploit the meaning of words other than that as provided by the syntax, in order to formulate a meaningful query. This turns out to be cumbersome, especially when advanced terminologies and large vocabularies should be used. This is also strengthened by the non-unique name assumption characterizing the application domain vocabulary. In this paper, we present a query construction paradigm which underlies the Meaning Driven Data Query Language $MDDQL$. It strongly relies on the construction of queries through suggestions of meaningful terms, in a preferred natural language, when requested by the end-user. An inference engine is responsible for the suggestion of a semantically consistent set, not only of application domain terms, but also of operator or operation terms, during the query construction process. All inferences are drawn at a "heuristic level" of representation of the vocabulary, i.e., on the basis of data structures (cyclic graph), and not at an "epistemological level", i.e., based on logic-like representations.

Keywords: Query Languages, Ontologies, Semantics, Information Retrieval.

1 Introduction

In this paper, we present a flexible query construction paradigm which underlies the Meaning Driven Data Query Language $MDDQL$ [13,14]. Flexibility and usability of the presented query construction mechanism become key issues, especially when advanced and/or large vocabularies within a multi-lingual user community are given. Furthermore, a *non-unique name assumption* underlies the given vocabulary, i.e., homonyms are allowed. It strongly relies on the construction of queries through suggestions of meaningful terms as presented in a preferred natural language, when requested by the end-user. An inference engine is responsible for the suggestion of a semantically consistent set, not only of application domain terms, but also of operator or operation terms, during the query construction process.

To this extent, the query construction process takes place in an interactive way where terms are rather suggested than anticipated by the end-user. For this purpose, a query construction blackboard is used where all interconnected query terms are placed. The suggestion of interpretable and semantically consistent query terms, i.e., compliant with the application domain semantics, can be activated on each query term currently appearing on the blackboard and, therefore, being part of the intended query. The query construction mechanism also copes with terms represented by the same natural language words but meaning different things (homonyms).

Given that the MDDQL vocabulary is a set of interrelated application domain terms, it is represented by an ontology-like structure, which reflects the semantics of the particular application domain. However, terms are conceived as *objects* rather than simple symbols. This allows the capture and representation of the *sense of words*, in addition to the meaning as reflected by the context of words which refers to their structural or syntax related properties. Therefore, it is possible to replace the texture of words with texture from other natural languages, while preserving the vocabulary structure. Thus it becomes possible to have the same query constructed with words in more than one natural language and still reflecting the same application domain semantics and referring to the same query results, since words are mapped to the data storage symbols.

According to the major distinction as given by McCarthy and Hayes [19] between logic-based representations, known as the ”epistemological level”, and the more procedural encoding, known as the ”heuristic level”, which introduces data structures for representing objects, the approach chosen for the representation of the vocabulary is closer to the heuristic level. This is, partly, due to the oversimplifying hypothesis that no symbolic notations such as those of logic, frames, production rules and semantic networks are able to capture the full richness of natural language [25], which in turn cannot capture to the full the richness of the world, as stated by Wittgenstein [26], and, partly, due to the injection of a hard core meaning, i.e., the *sense*, to the words which constitute the vocabulary. The latter is based on the simplifying hypothesis of De Saussure [22] concerning the irreversible relationship between *names* and *sense*.

Given the representation of the vocabulary at a heuristic level by means of a data structure such as a *cyclic graph* where the nodes stand for the objects representing the vocabulary terms, the inferences (suggestions) of terms are made on the basis of locating the appropriate nodes in the vocabulary graph, i.e., the node which corresponds to the term upon which a suggestion request has been posed, and the connected nodes (terms) in its neighborhood, according to a predefined distance and satisfaction of constraints. Thereby, the sense or hard core meaning of words is also taken into account during the query formulation.

The query construction mechanism also alleviates the task of formulating a query particularly in complex or advanced application domains, such as scientific applications, in that the end-user does not need to anticipate the vocabulary terms to be used or to make him/herself familiar with the whole domain vocabulary and its semantics, even if a *sub-language* is concerned such as medical terminology. Furthermore, there is no need to parse the query after its completion, in order to check, if the query is syntactically and semantically correct, in terms of compliance with the application domain semantics. Finally, there is no need to provide different parsing techniques according to the number of natural languages in which the vocabulary or query has been expressed.

Related work: The approach taken so far contrasts with other database specific query language or interfaces in that the language has been defined over vocabulary terms with semantic contents reflecting the current application domain and has a knowledge driven mechanism for the query formulation. In particular, database specific query languages are mainly designed for programmers and underlie a syntax formalism. Therefore, when end-users need to pose queries to a database, it is required that they have a substantial knowledge of syntax formalisms [10,5,18,21] and of the data storage model semantics in order to formulate queries.

Even easy-to-use languages such as SQL are intimidating for average users and require formal training to use. The problem becomes more acute when application domains are considered where a large number of parameters (attributes) and encoded values reflecting advanced terminologies are involved. Furthermore, quality aspects of attributes and/or values are not taken into account as a selection criterion for the inclusion of the term in the query.

In order to alleviate the query construction process by non-programmers, visual query formalisms [4,3] have been proposed in the past, which mainly rely on diagrammatic presentation of conceptual or database schemes. This technique, however, turns to be cumbersome, especially when very large database schemes, in terms of their basic constructs such as relations or classes as well as the amount of attributes, are addressed. Furthermore, one needs to be familiar with the semantics of the visual formalism and/or the interpretation of the encoding chosen for the realization of a conceptual schema and the data in a database. The problem becomes more crucial when complex or advanced terminologies are involved in the querying process. Even in cases such as [9,28] where an incremental formulation of the final query is considered, no usage of terminological or application domain semantics is made.

Semantics have also been used as a user guiding mechanism in interactive query formulation techniques. They are either database schema bound, in order to provide incremental or associative query answering [28,29], or they are ontology driven [24,20, 2]. The goal of [28,29] is to provide an end-user with context-sensitive assistance based on database modeling and probabilistic reasoning techniques combined with linguistic-based ones. To this extend, query formulation takes place in terms either of query completion for incomplete queries or suggestions of further predefined queries to reach a complex query goal [7,8]. To this extent, only that part of semantics which refers to the conceptual model is taken into account. There is no exploitation of the hard core meaning of words and/or quality aspects for the construction of the query.

On the other side, an ontology (knowledge) driven mechanism is used for the construction of queries [24,20,2], or as a dictionary for natural language processing [16, 15]. This approach enables the exploitation of the semantics of terms as given by their classification and taxonomic relationships. However, the main focus has been the exploration of taxonomies or classification structures as a context of meaning of words rather than the specification of a high-level query language with the expressive power as known by conventional database specific query languages, e.g., expression of logical, comparison or statistical operators. Moreover, since these approaches rely on conceptual structures organized around *names of terms* and not *terms conceived as objects*, it is quite difficult or even impossible to represent, and, therefore, exploit more than conceptual model bound semantics, such as conditional statements about the quality of values or attributes and their interpretation, or even to represent homonyms. This might lead to the construction of meaningless queries, since they might include terms which are relevant to particular users only, or they are mutually exclusive within the same query.

Finally, natural language (NLP) based query interfaces such as those provided by query answering systems or acting as query interfaces to databases [1,11,12], turn out to be tedious. This is due to the fact that it is a cumbersome task to lexically and semantically analyse the submitted query [23,27], especially when a multi-lingual user community is concerned. Therefore, the more natural languages are needed for representation of the query terms, the more lexical and semantic analyzers are needed (one for each natural language). Furthermore, completion of a query presupposes considerable knowledge of the vocabulary terms and their meaning, a task which is inefficient for the end-user or even impossible, especially when advanced and/or large terminologies are used.

In addition, despite the fact that the mainstream in linguistics stopped neglecting semantics with its relationship to knowledge representation after Noam Chomsky [6], in 1957, revolutionized the study of syntax and diverted attention away from semantics, as well as that, after 1974, there was a more integrated approach to both syntax and semantics (Montague and a number of other researchers made syntax subordinate to logic-based semantics), the main trend in syntactical and lexical analyzers has been the assignment of semantic roles to syntactic constituents as a fundamental aspect of NLP.

Given that a query in MDDQL is constructed by means of application domain semantics, which determine the syntax of the completed query, it is the semantics which drives the syntax as far as a completed query is concerned. The grammar of the query language is implicitly represented by the data structure underlying the representation of the vocabulary and, therefore, all potential queries that can be constructed. However, since this structure reflects as far as possible the application domain semantics, it can only lead, in most cases, to meaningful queries.

Organization of the paper: Section 2 gives an overview of the components of the $MDDQL$ query language. In order to illustrate the approach, a query construction paradigm is presented in section 3, with respect to the usage of the system for the needs of a clinical study of the treatment of myocardial infarctions in Swiss hospitals. In section 4, the $MDDQL$ inference mechanism is described, which is responsible for the suggestions of meaningful terms, in order to complete the construction of the query.

2 The Query Language Components

Following components constitute the MDDQL query language as a system:

1. The *MDDQL query construction blackboard* used as the user interface. The query is being assembled through user/system interaction in terms of posed suggestion requests over menu options as activated upon a particular query term such as *hospitalization*. The query, at any time of its construction, takes the form of an MDDQL query tree. It is this data structure which is finally submitted for interpretation and transformation. Since terms are internally represented as objects carrying on further attributes, the nodes of the query tree are also represented as objects.
2. The *ontology-like* representation platform for the MDDQL vocabulary together with the constraints on their use as bound to the application domain semantics.
3. The *MDDQL inference engine* in order to respond to the user's requests for meaningful suggestions of query terms during query construction. The inference engine is contacted each time suggestions for the consideration of semantically consistent subsets of query terms are requested by the end-user through the menu options, as mentioned above, in order to refine the query. The inferences rely on both the application domain semantics as represented by the *ontology* and the current set of already considered query terms.
4. The *MDDQL query tree interpretation and transformation component*[1], where each submitted query in terms of an MDDQL query tree is being traversed and transformed into an SQL-query.

3 A Query Construction Paradigm

In the following, we will try to illustrate the major underpinnings of the query construction mechanism which relies on the exploitation of the application domain semantics for making suggestions to the end-user. Since we do not expect that the end-user should be familiar with the semantics of conceptual or storage models, complex visual formalisms and advanced vocabularies, in order to construct a query, she/he is guided by a container of the first three components as described in the previous section and running on the client's site. In particular, these components participate in the query construction process by having the user selecting application domain terms and/or operators from suggested sets of terms. In order to illustrate the query construction paradigm, we will refer to the example as depicted in figure 1.

The user has the possibility of starting the query construction process with a term as selected from a list of initial terms which appears on the left panel. The list includes those terms which correspond to names of entity sets characterizing the application domain. Alternatively, a searching mechanism is also available in order to locate the requested term within the given vocabulary, in order to start the query construction with. Since searching is done on the basis of a *non-unique name assumption*, i.e., *homonyms* are allowed, each relevant term is included in the answer together with its context in terms

[1] This extends the scope of this paper and will not be presented here

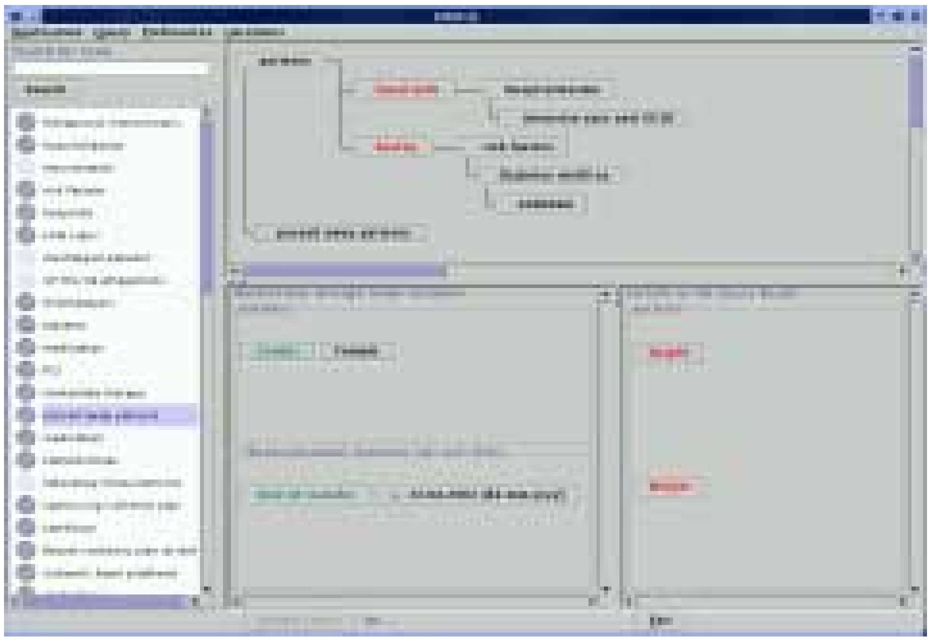


Fig. 1. The MDDQL blackboard with an example of a constructed query

of its *neighbored terms* such that the user can distinguish among homonyms on the basis of their meanings.

For instance, given the application domain *treatment of acute myocardial infarctions in hospitals*, the terms *Hospitals*, *Immediate therapy*, *Risk factors*, *Patients*, *Medication* are names standing for concepts usually representing entity sets with *Medication* and *Hospitals* having been defined more than once in the same vocabulary. Therefore, consideration of terms is bound to the underlying context, i.e., the hard core meaning of words such as annotations, condition of measurements, measurement units as well as their connections to other words in the neighborhood as provided by the structural properties of the vocabulary data structure.

For example, the user can distinguish between *Medication as Immediate Therapy* and *Medication given to Discharged Patients* by having the system exploiting and, therefore, responding as based on the structural properties of *Medication*. This means that the system is aware of *Medication* being part of *Immediate therapy* as well as that it is related to *Discharged patients* through the term (relationship) *given to*.

Having selected an initial term, for instance *Patients*, the term is placed on the top panel. From now on, it can be further refined by three possible kinds of restrictions (an example is depicted in figure 2), which are expressed by

Fig. 2. An example of suggestions of potential term (Patients) restrictions

- *predications* which are dynamically constructed by connecting mostly *verbs* standing for relationships and names of related concepts such as *having received* (the relationship) *immediate therapy* (the related concept),
- *specializations* of generic terms such as *discharged* or *passed away*,
- *value based restrictions* as posed on characteristic properties of the affected concept such as *height*, *weight* or *gender*.

All sets of suggested terms are presented to the end-user as a consequence for the activation of the menu option *constraints* from the menu popped up over the labeled box *patients*. Notice that this applies to all terms represented as labeled boxes on the query construction blackboard.

To this extent, there are two kinds of connecting links for the interpretation of connected terms on the top panel: a) *predication links* which connect *subjects* and the assigned *predications*, such as *patients having received immediate therapy*, b) *generalization links* which connect generic terms to more specific ones, such as *patients* and *passed away patients*. Notice also that, since links can be classified in the vocabulary in different ways, it is possible to distinguish among the different natures of generalization links such as *natural kinds*, *part of*, *constitutes of*, etc.

Assuming that the predications *faced with hospitalization* and *having risk factors* have been selected as a kind of restriction of the concept term *patients* and the more specific term *passed away patients* as a kind of restriction for the same concept (see also figure 1), further restrictions can be posed over the terms which constitute the restricted or refined query. For instance, the term *hospitalization* can be further restricted by the more specific term *intensive care unit (ICU)*, which is conceived as *part of* hospitalization.

The same holds for the restriction of the term *risk factors* through *diabetes mellitus*, since it is *a kind of* a risk factor. Note that potential restrictions are suggested, only if there are any discovered.

Value or instance based restrictions can be expressed when characteristic properties such as *Gender* or *Date of transfer* have been selected from the set of suggested restrictions as proposed for the concept terms *patients* (figure 2) and *intensive care unit (ICU)*, respectively. Each of these properties also appear as labeled boxes on the panel on the left corner (figure 1), from which values and/or comparison operators can be assigned. However, the set of comparison operators suggested upon request (default operator is *equals*) depends on the nature of the property. Therefore, the comparison operator *less than* is suggested only for *Date of transfer*, since *Gender* is recognized as a *categorical variable*, i.e., it does not make sense to apply such a kind of comparison operator for values to be assigned to the property *Gender*.

The latter gives emphasis to the *meaningful assignment of operators* in the query language. Therefore, only semantically consistent operators are suggested according to the nature of attributes or variables. This also applies to the class of statistical operators such as *average*, *maximum*, *minimum*, etc., which can be suggested as applicable functions for properties such as *Height* but not to *Gender*, since the latter is recognized as a categorical variable.

Logical operators other than *AND*, which is the default operator, can be assigned to all terms which are related to both *entity set* and *value based* restrictions. For example, the *NOT* operator has been assigned to the term *having* (figure 1) indicating the intention to address in the query result only patients who have not reported "unknown diabetes mellitus" as kind of "risk factor". Similarly, a *negation* of the term *intensive care unit* would have indicated the fact that the user is interested in those patients, who have been hospitalized but not been in the intensive care unit during the hospitalization.

Notice also that inverse relationships among terms are also taken into account. For example, assuming that the user would have started the query construction process with the term *Immediate therapy*, the restriction through the predication *applied to patients* will be suggested, where *applied to* is known in the vocabulary as an inverse term of *having received*. However, the set of suggested predications over the term *patients* will not include *having received immediate therapy*, since this leads to cycles in the query, i.e., back to the concept term *Immediate therapy*. To this extent, *mutually exclusive terms* within the same query are prevented.

Further semantic contradictions are detected over terms such as *done* and *unknown* when they are expressed as AND-connected restrictions (such a combination always lead to empty sets in the query result). Accordingly, the inclusion of *passed away patients* and *discharged patients* in the same query, as AND-connected specific terms of *patients*, is rejected as meaningless, since, according to the application domain semantics, it will be never possible that a passed away patient has also been discharged.

Summarizing, suggestions of terms made by the $MDDQL$ query construction mechanism comply with both definitions of the vocabulary terms and their constraints on use. Both reflect, to some extent, the semantics of the application domain. The vocabulary is extended by terms referring to *operations* or *operators*. In the following, we will have a closer look at the $MDDQL$ inference mechanism.

4 The MDDQL Inference Mechanism

The inference engine runs as part of the *application client container* together with the MDDQL query construction blackboard. All suggestion requests posed by the end-user through the menu options, as activated upon the terms appearing on the MDDQL query construction blackboard, lead to activation of *methods* or *services* provided by the MDDQL inference engine. However, *meaningful terms* to be further considered for the completion of the query are drawn on the basis of two major data structures: a) the $MDDQL$ *query tree* which accommodates all terms currently constituting an intended query, b) the *cyclic graph* which expresses, to some extent, the application domain semantics in terms of vocabulary terms and constraints on their use.

All nodes on both data structures representing vocabulary or query terms are conceived as *objects*. This makes terms having instance variables such as a term unique identifier, a natural language word, the corresponding implementation symbol, an annotation, etc. Since both structures rely upon *term unique identifiers* rather that *words*, it is feasible to express texture such as natural language words and/or annotation in more than one natural language, whereas the structure is preserved. Having said that, both the immediate expression of a particular query in terms of the query tree and the expression of the domain vocabulary can be accomplished by using terminologies from different languages as far as structure and not contents of term nodes are concerned. Finally, it is the query tree which gets transformed toward database specific query languages such as SQL and, therefore, enables receiving the same query result, even if the query has been expressed by words in different natural languages.

Each time one or more terms are added and, therefore, extend the current query, the $MDDQL$ query tree gets manipulated by the inference engine. Furthermore, the query tree is bound to an *environment* such as the targeting *location*, for instance, name of the hospital, to which a particular query should apply. In this sense, the inference engine is always aware of the current stage (context) of a query in terms of both query language terms and environment. In the following, we will have a closer look at the definition of the underlying data structures.

The $MDDQL$ query tree: It is defined as a set of interconnected *query term nodes* which are represented as objects and carry on all semantic information necessary for the inferences for further terms as well as the interpretation of the query tree (a submitted query) toward the generation of a database specific query language such as SQL. Operators or operations are also represented as objects. However, they are assigned to the query term nodes rather than being tree nodes themselves.

The query tree consists of nodes which have been classified either as *Entity Set*, or *Relationship*, or *Property*, or *Value* node. The structure of an $MDDQL$ query tree, however, underlies some constraints as far as this classification schema is concerned.

- The root of the query tree is always an *Entity Set* term node.
- An *Entity Set* term node might have as children other *Entity Set* term nodes or *Relationship* term nodes.
- A *Relationship* term node might have as children other *Relationship* term nodes or *Entity Set* term nodes.

– Each *Entity Set* or *Relationship* term node might have as children *Property* term nodes.
– Each *Property* term node might have as children other *Property* term nodes or *Value* term nodes.
– A *Value* term node might have as children other *Value* nodes.

The vocabulary cyclic graph: It provides a *semantic space* where all application domain terms with their constraints on use are expressed. Despite the fact that *Ontological Engineering* is still a relatively immature discipline, some methodologies and criteria of analyzing them can be found in [17]. To this extent, the ontological engineering approach taken so far is closer to the SENSUS-based methodology which aims at the structuring and representation of conceptual structures to be used as a dictionary for natural language processing [16,15]. However, one of the major differences is that conceptual structures are organized around *names of terms* and not around *objects* as it is the case for the $MDDQL$ vocabulary.

The semantic space of the application domain terms consists of atomic terms connected with links which either lead to hierarchical structures of terms or to attributive structures. Links which constitute hierarchical structures of terms are classified according to the semantics of the linkage such as *is-a, part-of, constitutes-of*, etc. For instance, *Thrombolysis*, *PTCA*, *Antagonists* and *Medication* are connected hierarchically with the term *Immediate therapy* through *is-a* links. On the other hand, *Date of thrombolysis*, *Reasoning for denial of thrombolysis* and *applied to* are connected with the term *Thrombolysis* through attributive links. The term *applied to* is further connected with the term *patients* with an attributive link too.

From another perspective point of view, the organization of the semantic space of terms can be conceived as a multi-dimensional conceptual space where each dimension or axis stands for an atomic term. The points on a particular axis indicate the terms with which the dimensional term is connected. From each point on an axis, there are outgoing axes, which further specify a particular term. For instance, the axis for *Thrombolysis* starts at a point on the axis for *Immediate therapy*. However, constraints are also defined for the validity of terms and, consequently, on their use, i.e., the existence of an axis within the semantic space is not static but rather dynamic.

These constraints might refer to application domain semantics such as *adaptable value domains*, *mutually exclusive terms* in case of AND-connected restrictions within a query, or location specific constraints. These constraints might have an impact on the relativeness of attributes and value domains, since their validity, i.e., suggestion is bound to the current query. For example, the attribute *Troponin I* as a *laboratory measurement* is not suggested, if the target location for the query is the *Triemli* hospital, since there is no such measurement available there. The expressed constraints might also lead to the rejection of a query as meaningless, if it includes *mutually exclusive terms* such that they always lead to empty sets in the query result. For example, the consideration of *passed away* AND *discharged patients* always lead to empty sets, since, according to the definition of the application domain, it is not possible to have a patient who passed away and has been discharged.

The vocabulary also includes *operational terms* which underlie a classification such as *comparison*, or *logical*, or *statistical operator*. There also constraints holding on their use which express the conditions under which (sub)sets of operators will be suggested to the user. This depends upon the nature of the application domain terms which have already been considered within the query under construction. In the following, we give an overall description of the inference mechanism, which is responsible for the suggestion of meaningful terms. This mechanism operates upon the two major data structures as described in this section.

4.1 Which Terms or Queries Are Meaningful?

In order to infer which terms make sense to be included in the set of proposed terms, two major steps need to be taken: 1) the location of the term within the $MDDQL$ vocabulary structure (the cyclic graph), upon which a suggestion request has been activated on the blackboard, 2) determination of the context of the located term. The latter is determined in terms of a) the neighborhood of the term under consideration as defined by a distance measure, b) the constraints which might hold for the inferred neighboring terms.

Notice that this apply with slight differences to both *query extension* and *term search engine* facilities as provided by the MDDQL system. The latter enables a *non-unique name* based search of a particular term given that the answer is provided within neighbored terms which constitute the context of the searched term. The search mechanism is particularly used when an initial term needs to be located with which we would like to start the construction of the query.

Since location of the related terms as well as the neighbored terms and their constraints are given in terms of unique term identifiers only, terms expressed with same words but having different meanings can be taken into consideration. The validity of constraints, however, can be checked either at the time of the inference of meaningful terms or at the time of submission of the query. This holds, for example, when mutually exclusive terms need to be detected when the query has been completed. Summarizing, the following kinds of semantic contradictions are detected and, therefore, the following sets of terms might be excluded from the current set of neighbored terms:

- Predication which are already in the $MDDQL$ query tree, i.e., the query under construction.
- Predications which are based upon relationship terms which have an inverse term already appearing within the $MDDQL$ query tree. This avoids cycles which might be built between terms connected by verbs in both directions.
- Entity sets which contradict with entity sets to which they are connected through the logical operator *AND* within the query. For instance, *Discharged* and *Passed away* as sub-categories of *Patients* which are not semantically consistent within the same query, since a patient either survived the hospitalization or passed away. The same holds for values such as *done* AND *unknown* within the same set of suggested value terms.
- Property terms to be used in conditional clauses which are only valid under certain circumstances. For instance, *Reasoning for denial of thrombolysis* is only valid, if *Thrombolysis* has not been used as restriction indicating that we are interested in patients who received thrombolysis as a kind of immediate therapy. Validity of properties is also checked upon the targeting location (hospital name) of the query.

– Operator terms which do not comply with the nature of the term under consideration. For instance, if the term is a value assigned to a conditioning property, then only the set of *comparison operators* will be suggested. Furthermore, if it happens that the conditioning property is a *Categorical Variable*, then the subset of comparison operators $\{>, <, <=, >=\}$ will be excluded from the set of suggested operators as being meaningless.

Conclusion

A flexible query construction mechanism of, as meaningful as possible, queries has been presented in this paper. This underlies the specification of $MDDQL$ as a high-level, end-user oriented query language. It mainly relies upon system made suggestions of meaningful query terms to the end-user in order, in order to complete an intended query, in terms of application domain semantics compliant terms. Therefore, there is no need to have the user typing syntactically correct queries which are either prone to syntactical mistakes or might be meaningless, i.e., inconsistent with the application domain semantics and/or terminology, as it is mostly the case in NL based query interfaces.

In addition, there is no need to have the end-users making themselves familiar with visual query language formalisms which also lack data interpretation mechanisms or multi-lingual expressiveness. Furthermore, a considerable knowledge of the application domain terminology is expected from the end-user, in order to formulate a query. It also turns out to be a cumbersome task having various parsing mechanisms, depending on the number of natural languages in which a potential query might be constructed.

The approach alleviates the task of formulating a query, especially when large database schemes are addressed and/or advanced terminologies are used. This is also strengthened by the fact that there is no need of learning and using a particular syntax of a query language. This kind of interactive query construction, with the query tree reflecting the query under construction, also avoids the need of lexically analyzing a query, as known by natural language based query interfaces in query answering systems. To some extent, the application domain semantics drive the construction of the query syntax and not vice versa.

Furthermore, $MDDQL$ relies on a *name centric* way of dealing with terms such that *homonyms* are allowed, since disambiguation over their meaning is enabled through the accompanying context (neighbored terms). All vocabulary terms can be expressed and presented to the user with their assigned natural words in a preferred natural language. This allows the addressing of the same query result, even if the query has been expressed by using more than one natural language for the domain terminology.

References

1. N.R. Adam and A. Gangopadhyay. A form-based natural language front-end to a CIM database. *IEEE Transactions on Knowledge and Data Engineering*, 9(2):238–250, April 1997.
2. S. Bechhofer, R. Stevens, G. Ng, A. Jacoby, and C. Goble. Guiding the User: An Ontology Driven Interface. In Norman W. Paton and Tony Griffiths, editors, *Proc. User Interfaces to Data Intensive Systems (UIDIS99)*, pages 158–161, Edinburgh, September 1999. IEEE Press.

3. J. Cardiff, T. Catarci, and G. Santucci. Semantic query processing in the VENUS environment. *International Journal of Cooperative Information Systems*, 6(2):151–192, June 1997.
4. T. Catarci, M.F. Costabile, S. Levialdi, and C. Batini. Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260, April 1997.
5. R.G.G. Catell and D. K. Barry, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann Publishers, Inc., 1997.
6. Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.
7. Wesley W. Chu, Hua Yang, Kuorong Chiang, Michael Minock, Gladys Chow, and Chris Larson. CoBase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 1996.
8. W.W. Chu and Q. Chen. A Structured Approach for Cooperative Query Answering. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):738–749, October 1994.
9. Ignacio Gil, Peter M.D. Gray, and Graham J.L. Kemp. A Visual Interface and Navigator for the P/FDM Object Database. In N. Paton and T. Griffiths, editors, *Proc. of the 1st Inter. Workshop on User Interfaces to Data Intensive Systems, (UIDIS 99)*, pages 54–63, Edinburgh, Scotland, September 1999. IEEE Computer Society Press.
10. J. R. Groff and P. N. Weinberg. *SQL : the complete reference*. Osborne/McGraw-Hill, Berkeley, 1999.
11. B.J. Grosz, D. Appelt, P. Martin, and F. Pereira. TEAM: An experiment in the design of transportable natural-language interfaces. *Journal of Intelligent Information Systems*, 32(2):173–244, 1987.
12. G.G. Hendrix, E.D. Sacerdoti, D. Sagalowicz, and J. Slocum. Developing a natural language interface to complex data. *ACM Transactions on Database Systems*, 3(2):105–147, June 1978.
13. E. Kapetanios, D. Baer, P. Groenewoud, and P. Mueller. The Design and Implementation of a Meaning Driven Data Query Language. In Jessie Kennedy, editor, *Proc. 14th Inter. Conf. on Scientific and Statistical Databases*, Edinburgh, Scotland, July 2002. IEEE Computer Society Press.
14. E. Kapetanios, M.C. Norrie, and D. Fuhrer-Stakic. MDDQL: A Visual Query Language for Meta-data Driven Querying. In *5th IFIP Inter. Working Conf. on Visual Database Systems*, Fukuoka, Japan, May 2000. Kluwer Academic Publisher.
15. K. Knight, I. Chancer, M. Haines, V. Hatzivassiloglou, and E.-H. Hovy. Filling Knowledge Gaps in a Broad-Coverage MT System. In *Proc. of the 14th IJCAI-94*, Montreal, Canada, 1995.
16. K. Knight and S. Luck. Building a Large Knowledge for Machine Translation. In *Proc. of the AAAI-94*, Seattle, USA, 1994.
17. Mario Fernadez Lopez. Overview of Methodologies for Building Ontologies. In *Proc. of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, Stockholm, Sweden, August 1999.
18. David Maier. Database Desiderata for an XML Query Language. In *Proc. of the Query Languages Workshop*. Cambridge, Mass., December 1998.
19. J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In D. Michie and B. Meltzer, editors, *Machine Intelligence*, volume 4. Wiley, New York, 1969.
20. N.W. Paton, R. Stevens, P.G. Baker, C.A. Goble, S. Bechhofer, and A. Brass. Query Processing in the TAMBIS Bioinformatics Source Integration System. In *Proc. 11th Int. Conf. on Scientific and Statistical Databases (SSDBM)*, pages 138–147 1999. IEEE Press, 1999.
21. J. Robie, J. Lapp, and D. Schach. XML Query Language (XQL). In *Proc. of the Query Languages workshop*, Cambridge, Mass., December 1998.
22. F. De Saussure. *Cours de Lingistique Generale*. Tullio de Mauro, Paris: Payot, 1972.
23. R. Schank. Identification of conceptualizations underlying natural language. In *Computer Models of Thought and Language*, pages 187–247. Freeman, San Francisco, CA, 1973.

24. R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N.W. Paton, and C.A. Goble. TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. *Bioinformatics*, 16(2):184–186, 2000.
25. S. Ullman. *Semantics - An Introduction to the Science of Meaning*. Blackwell, Oxford, 1962.
26. L. Wittgestein. *Philosophical Investigations*. Blackwell, Oxford:Basil, 1953.
27. W. A. Woods. Transition network grammars for natural language analysis. *CACM*, 13(10):591–606, Oct 1970.
28. G. Zhang, W. W. Chu, F. Meng, and G. Kong. Query Formulation from High-Level Concepts for Relational Databases. In N. Paton and T. Griffiths, editors, *Proc. User Interfaces to Data Intensive Systems, UIDIS 99*, pages 64–74, Edinburgh, Scotland, September 1999. IEEE Computer Society Press.
29. Guogen Zhang. *Interactive Query Formulation Techniques for Databases*. PhD thesis, University of California, Los Angeles, 1998.

An Augmented Visual Query Mechanism for Finding Patterns in Time Series Data

Eamonn Keogh[1], Harry Hochheiser[2], and Ben Shneiderman[2,3]

[1]Computer Science & Engineering Department
University of California - Riverside
Riverside, CA 92521
eamonn@cs.ucr.edu
[2]Department of Computer Science
Human-Computer Interaction Lab
[3]Institute for Advanced Computer Studies, and Institute for Systems Research
University of Maryland
College Park, MD 20742 USA
{hsh,ben@cs.umd.edu}

Abstract. Relatively few query tools exist for data exploration and pattern identification in time series data sets. In previous work we introduced *Timeboxes*. Timeboxes are rectangular, direct-manipulation queries for studying time-series datasets. We demonstrated how Timeboxes can be used to support interactive exploration via dynamic queries, along with overviews of query results and drag-and-drop support for query-by-example. In this paper, we extend our work by introducing Variable Time Timeboxes (VTT). VTTs are a natural generalization of Timeboxes, which permit the specification of queries that allow a degree of uncertainty in the time axis. We carefully motivate the need for these more expressive queries, and demonstrate the utility of our approach on several data sets.

1 Introduction

Time series data sets are ubiquitous, appearing in many domains including finance, meteorology, physiology and genetics. To date, most information visualization work on these data sets has focused on display and interactive exploration, often emphasizing the periodic nature of some calendar-based data sets [6]. Work in data mining has addressed the need for additional tools to identify patterns of trends of interest in these data sets. Algorithmic and statistical methods for identifying patterns [1,2,3,5,8,11] have provided substantial functionality in a wide variety of situations. In domains such as stock price analysis, familiar patterns have been named and identified as shorthand approaches to identifying trends of interest [12]. Tools for specifying dynamic queries over these data sets have recently been developed: QuerySketch supports query-by-example based on a sketch of a desired profile [20], and Spotfire's Array Explorer 3 supports graphical queries for temporal patterns [18].

In previous work we introduced timeboxes: an interactive mechanism for specifying queries on temporal data sets. Timeboxes are rectangular regions that are placed and directly manipulated on a timeline, with the boundaries of the region providing the relevant query parameters. In this paper we introduce an extension to timeboxes, which allow queries that have some flexibility in the time axis. Researchers in speech processing and other fields have long known the utility of such "time warped" queries. We call our new approach Variable Time Timeboxes (VTT). We carefully motivate the need for such queries, and demonstrate the utility of our approach on several data sets.

The rest of this paper is organized as follows, in Section 2 we provide an extensive review of Timeboxes, and introduce TimeSearcher, an application that uses timeboxes to provide an interactive environment for visualizing and querying time series. In Section 3 we motivate, introduce and test our new VTT approach. Section 4 considers related work. Finally, in Section 5 we offer some conclusions and directions for future work.

2 Timeboxes: Interactive Temporal Queries

Timeboxes are rectangular query regions drawn directly on a two-dimensional display of temporal data. The extent of the Timebox on the time (x) axis specifies the time period of interest, while the extent on the value (y) axis specifies a constraint on the range of values of interest in the given time period. More concretely, we define a timebox as follows.

Definition 1: A *timebox,* defined by two points (x_1, y_1) and (x_2, y_2), is a constraint on a time series indicating that for the time range $x_1 \le x \le x_2$, the dynamic variable must have a value in the range $y_1 \le y \le y_2$, (assuming $\mathrm{y}_2 \ge \mathrm{y}_1$).

Multiple timeboxes can be drawn to specify conjunctive queries. Data sets must match all of the constraints implied by the timeboxes in order to be included in the result set.

Creation of timeboxes is straightforward: the user simply clicks on the desired starting point of the timebox and drags the pointer to the desired location of the opposite corner. As this is identical to the mechanism used for creating rectangles in widely used drawing programs, this operation should be familiar to most users. Once the timebox is created, it may be dragged to a new location or resized via appropriate resize handles on the corners, using similarly familiar interactions.

Query processing occurs on mouse-up. When the user releases the mouse, the current position of the timebox is stored, the query is updated, and the new result set is displayed.

Construction of timeboxes is aided by drawing all of the items in the data set directly on the query area. This "graph envelope" display provides additional insight into the density, distributions, and patterns of change found among items in the data set, in a display that is reminiscent to a parallel coordinates visualization [10] (Figure 1).

The example data set shown in Figure 1 contains weekly stock prices for 1430 stocks and will be used in a brief scenario to illustrate the use of timeboxes. An analyst interested

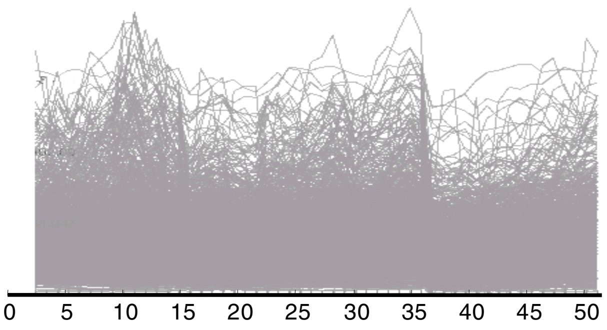


Fig 1. A "graph envelope" overview, formed by superimposing the time series for all of the items in the data set

in finding stocks that rose and then fell within a four-month period might start by drawing a timebox specifying stocks that traded between \$28 and \$64 during the first few weeks. When this query is executed, the graph envelope is updated to show only those records that match these constraints. We can quickly see that this query substantially limits the number of items under consideration, but many still remain (Figure 2.A).

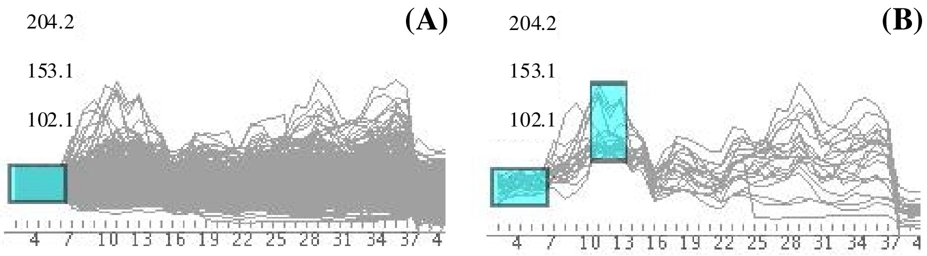


Fig. 2. (**A**) A single timebox query, for items between \$28 and \$64 during weeks 1-5. (**B**) A refinement of the query in (**A**) reduces the number of matching time series

To find stocks in this set that rose in subsequent weeks, the user draws a second box, specifying items that traded between \$73 and \$147 in weeks 10-12 (Figure 2.B).

As timeboxes are added to the query, the graph envelope provides an ongoing display of the effects of each action and an overview of the result set. Once created, the timeboxes can be scaled or moved singly or together to modify the query constraints.

The use of simple, familiar idioms for creation and modification of timeboxes supports interactive use with minimal cognitive overhead. Rapid automatic query processing (<100 ms) on mouse-up events provides the fast response necessary for dynamic queries [16], thus supporting interactive data exploration. Users can easily and quickly try a wide range of queries, and modify them to quickly see the effects of changes in query parameters. This ability to easily explore the data is helpful in identifying specific patterns of interest, as well as in gaining understanding of the data set as a whole.

Timeboxes also differ from traditional dynamic query widgets [16] in their construction and manipulation directly on the data space. As timeboxes are drawn directly on a graph space suitable for plotting a time series, the queries are easily interpreted at a glance.

2.1 TimeSearcher

TimeSearcher [9] uses timeboxes to pose queries over a set of entities with one or more time-varying attributes. Entities have one or more static attributes, and one or more time-varying attributes, with the number of time points and the definition of those points being the same for every entity in a given data set. If there are multiple time-varying attributes, any one of them can be selected for querying, through a drop-down menu that specifies the dynamic attribute being queried. All active queries refer to the same attribute.

When a data set is loaded, entities in the data set are displayed in a window in the upper left-hand corner of the application. Each entity is labeled with its name, and the values of the active dynamic attribute are plotted in a line graph. Complete details about the entity (details-on-demand) can be retrieved by simply clicking on the graph for the desired entity: this will cause the relevant information to be displayed in the lower right-hand window (Figure 3).

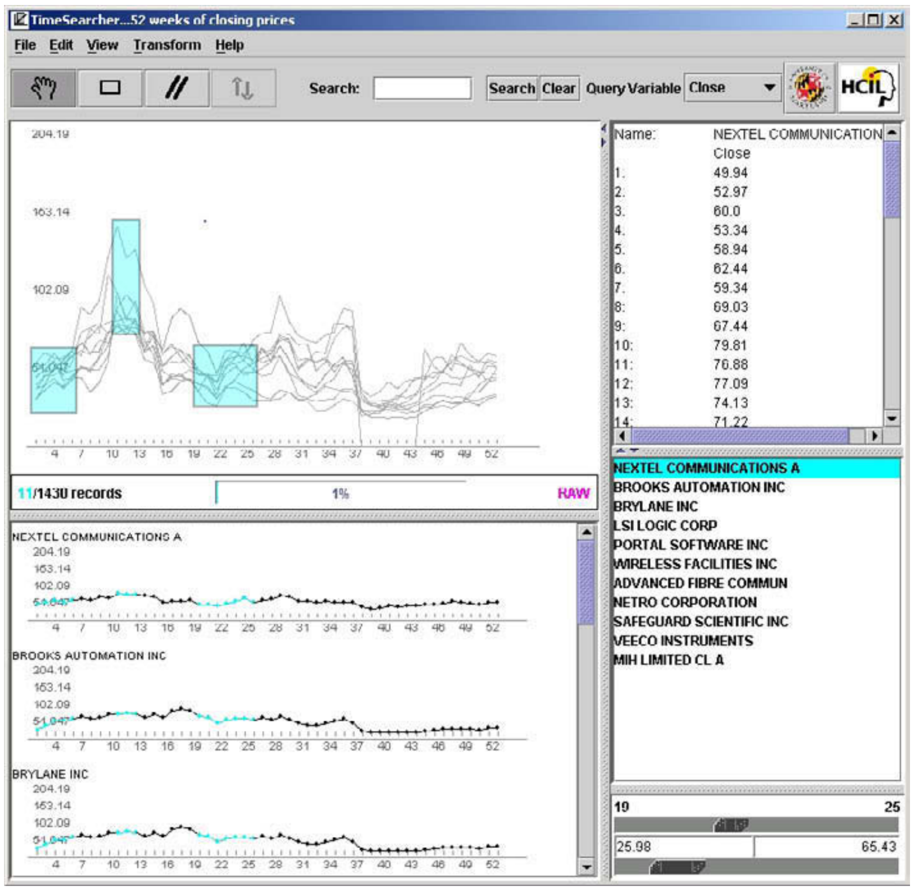
The upper-left corner of the TimeSearcher window is the query input space. This space initially contains an empty grid. To specify a query, users simply draw a timebox in the desired location. Query processing begins as soon as users release the mouse, signifying the completion of the box. Thus, users do not need to press a button to explicitly start a search. When query processing completes, the display in the top half of the application window is updated to show those entities that match the query constraints. For all of these entities, the time points that correspond to the queries are highlighted, in order to simplify interpretation of the display.

Once the initial query is created, the timeboxes can be moved and resized. The hand and box icons on the lower toolbar are used to switch between creating timeboxes and moving/resizing them. As is the case with initial timebox creation, query processing begins immediately upon completion of the movement/resizing of the timebox.

When multiple timeboxes are present, they can be modified individually or simultaneously in groups of two or more. This functionality is particularly useful for searches for complex patterns (Figure 3). In these cases, users can select some or all of the timeboxes (using standard lasso and shift-click interactions) and simultaneously apply the same translation and/or scale along either or both axes to all selected timeboxes. This is useful for searching for instances of a pattern that vary slightly in scale or magnitudes, or for modifying queries based on example items.

TimeSearcher uses "Graph Envelope" displays to provide overviews of the entire data set [9], and a simple drag-and-drop "query-by-example" mechanism supports the similarity queries often discussed in research in the mining of time series data [1,3,5,8,11].

TimeSearcher is implemented in Java, using the Swing toolkit for user-interface components. Drawing and scenegraph control in the data and query displays is provided by Jazz, a zooming toolkit written in Java [4].


TimeSearcher...52 weeks of closing prices
File Edit View Transform Help
Search:
Search Clear Query Variable Close
HCIL
11/1430 records
1%
RAW
19 25
25.98 65.43

Name:	NEXTEL COMMUNICATION
	Close
1:	49.94
2:	52.97
3:	60.0
4:	53.34
5:	58.94
6:	62.44
7:	59.34
8:	69.03
9:	67.44
10:	79.81
11:	76.88
12:	77.09
13:	74.13
14:	71.22

NEXTEL COMMUNICATIONS A
BROOKS AUTOMATION INC
BRYLANE INC
LSI LOGIC CORP
PORTAL SOFTWARE INC
WIRELESS FACILITIES INC
ADVANCED FIBRE COMMUN
NETRO CORPORATION
SAFEGUARD SCIENTIFIC INC
VEECO INSTRUMENTS
MIH LIMITED CL A

Fig. 3. The TimeSearcher application window. Clockwise from upper-left: query space, details-on-demand, item list, range sliders for query adjustment, and data items

3 An Augmented Query Mechanism

Timeboxes offer a very flexible query language, but it is not complete. To see why, consider the following motivating example. One of the classic symptoms of Hodgkin's disease is the appearance of two dramatic elevations of the patient's temperature in a 24-hour period. Figure 4 shows two examples.

One notable feature of such patterns is that the two peaks may be close together (almost to the point of merging) or as far apart as 18 hours. This uncertainly in the time axis is impossible to represent with Timeboxes. If we place two Timeboxes six hours apart (the mean value reported in the literature), we run the risk of missing positive examples which where the peaks are further apart or closer together.

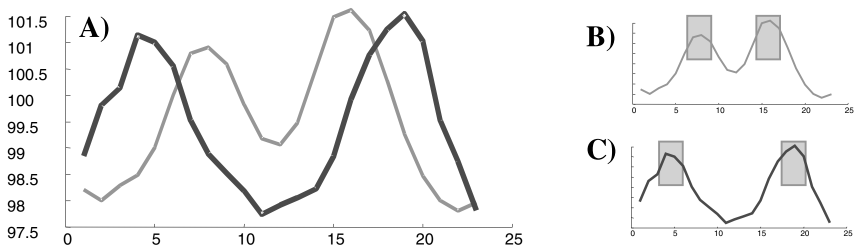


Fig. 4. Two peaks in a patients temperature over a single day is a classic symptom of Hodgkin's disease, however the peaks may be an arbitrary distance apart as shown in (A). Our current definition of TimeBoxes cannot detect peaks at arbitrary locations. Although we could create queries to find certain examples of double peak patterns (i.e B *or* C), we cannot use TimeBoxes to create a single query to which will discover all double peak patterns

Two overcome this limitation we can expand the definition of Timebox, to allow constraints of the following form. The time series of interest must be within a specified range for some specified duration *anytime* within a specified time window. We call such constraints, a Variable Time Timeboxes (VTT). We can define VTTs more concretely as follows.

Definition 2: A *Variable Time Timebox* (VTT), defined by two points (x_1, y_1) and (x_2, y_2) and a single integer R, is a constraint on a time series indicating that for the time range $x_1\text{-}R \le x \le x_2+R$, the dynamic variable must have a value in the range $y_1 \le y \le y_2$, for at least $(x_2\text{-}\ x_1)$ consecutive time units, (assuming $y_2 \ge y_1$ and $x_2 \ge x_1$).

Figure 5 illustrates the difference between Timeboxes and VTTs. Note that Timeboxes can be considered a special case of VTTs, where the parameter R is set to zero. As such we can claim that VTTs are more flexible and expressive than Timeboxes.

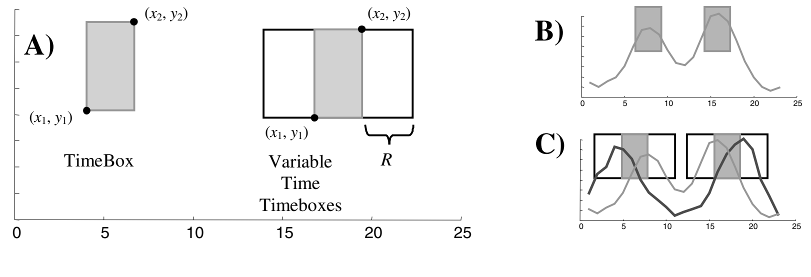


Fig. 5. A Timebox may be visualized as a shaded rectangle through which a qualifying sequence must pass. In contrast, an Variable Time Timeboxes (VTT) requires a qualifying sequence to have its value within the shaded region, allowing the shaded region to move anywhere within the larger constraining rectangle (**A**). Unlike Timeboxes (**B**), VTTS are able to detect patterns with a degree of uncertainty in the time axis

VTTs can be placed on the GUI the same way Timeboxes are. By default the *R*-value is set to zero, thus, on mouse-up they act as classic Timeboxes. However the user can left click the edges of the VTT to (symmetrically) resize the rectangle representing the R-value.

3.1 Efficiently Supporting VTTs

Although VTTS are more expressive than classic TimeBoxes, they also require more effort to support efficiently. As explained in previous work, Timebox queries can be processed via a modified orthogonal range tree query algorithm [7]. We can support VTTs efficiently by leveraging off previous work on indexing inverse time series queries [13]. An inverse query computes all time points at which a sequence contains values equal to the query value. A time series can be indexed by grouping sections with little variability in the Y-axis, and bounding the sections with a Minimum Bounding Rectangle (MBR). Figure 6 illustrates the idea.

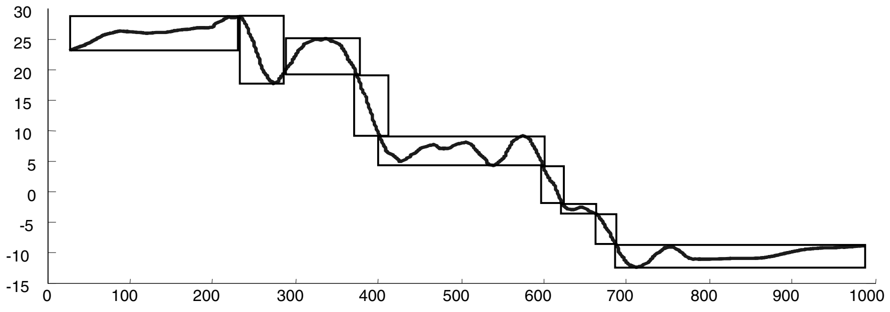


Fig. 6. A time series divided into 9 MBRs. Although the MBRs are actually one dimensional, they are shown as two dimensional for clarity

The MBRs are indexed using a R-Tree [8], a data structure that can efficiently support a variety of queries.

For a VTT query that asks for all time points that correspond to a time series that takes on a value equal to *y'*, where $y_1 \leq y' \leq y_2$, we perform a range query with the R-Tree to retrieve all the leaf queues that enclose value *y'*. The returned time series are guaranteed to contain the full answer set to the query, plus possibly some "false alarms", i.e. subsequences that on the value y', but for less than (x_2- x_1) consecutive time units. These false alarms are removed by a post-processing step. This technique is similar in spirit to the indexing technique introduced in [8]. Here the authors prove that any lower bounding technique can be used to index data, such that all qualifying sequences are retrieved. The only issue is the tightness of the lower bound. If the lower bound is very weak, many additional non-qualifying sequences will be retrieved (the so-called "false alarms"), although these can be removed in a post processing stage, this would cause the system to be degrade greatly in terms of speed. The other extreme, an arbitrarily tight lower bound, would allow a constant time access method. In general, this method works very well for most time series, since most real world time series have strong autocorrelation, and are therefore well approximated in by the low dimensionality MBRs, giving relatively tight bounds.

3.2 An Experiment to Demonstrate the Utility of VTTS

Our subjective evaluation of VTTs suggests that it is a useful tool for exploring large time series databases; in this section we provide an objective experimental evaluation of their utility. In particular, we will support our claim that VTTs allow more expressive queries, by showing experiments where our proposed approach outperforms previous work in the task of separating two different classes of time series.

In order to allow replication of our results, and to permit comparison to existing work, we have used the two most referenced times series datasets in the literature.

- **Cylinder-Bell-Funnel**: This synthetic dataset has been in the literature for 8 years, and has been cited at least a dozen times [21]. The dataset consists of 3 different basic shapes, which are produced by a function that has a stochastic element. For our experiments we consider only the "Funnel" and "Bell" classes.
- **Control-Chart**: This synthetic dataset has been freely available for the UCI Data Archive since June 1998 [21]. There are 6 different classes of time series. For our experiments we consider only the "Increasing Trend" and "Upward Shift" classes.

Our experiment consists of first showing the subject labeled examples of each class. We show the user as many as they wish to see, until they can identify unlabeled examples with 100% precision. The user is then shown a graph envelope view containing 10 examples of each of the two classes. The users task is to create a single query that can separate the two classes. The user attempts this with both Timeboxes and VTTS (with their choice of order). The experiment is repeated 10 times for each subject, and for each of the two datasets. Figure 7 shows an example of an experiment with the Control Chart dataset.

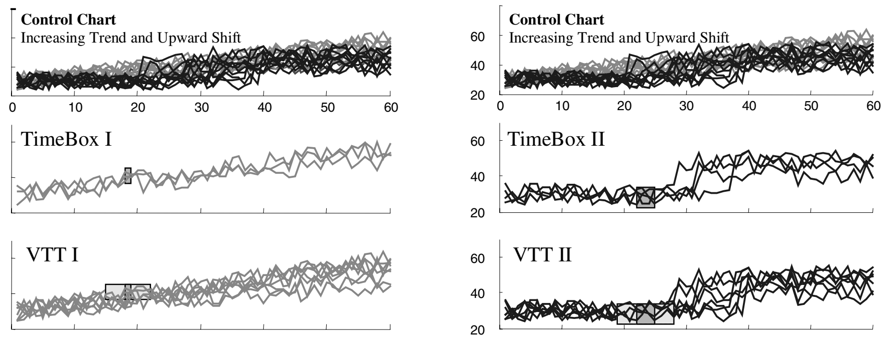


Fig. 7. The basic experiment setup on the Control Chart dataset. The user is shown a graph envelope containing 20 sequences, 10 each of "Increasing Trend" and "Upward Shift". The user is asked to create a single query to separate the two classes, using timeboxes and VVTs. The two queries may be placed at different locations, however for comparison purposes they are placed in exactly the same locations above. Note that for both cases VTTs are able to do a better job of separating the classes

In the example shown timeboxes separated out 3 of the 10 examples of "increasing trend", whereas VVTs were able to separate out 8 of the examples. For the second experiment we attempted to separate out just the "upward shift" sequences. Here timeboxes separated out 4 of the 10 examples of whereas VVTs were able to separate out 7 of the examples. Figure 8 depicts an example of an experiment with the Cylinder-Bell-Funnel dataset, where similar results can be observed.

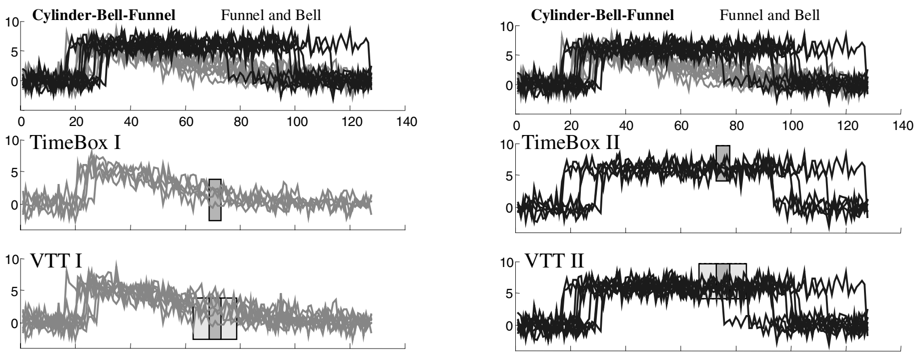


Fig. 8. The basic experiment setup on the Cylinder-Bell-Funnel dataset. The user is shown a graph envelope containing 20 sequences, 10 each of "Funnel" and "Bell". The user is asked to create a single query to separate the two classes, using timeboxes and VVTs. The two queries may be placed at different locations, however for comparison purposes they are placed in exactly the same locations above. Note that for both cases VTTs are able to do a better job of separating the classes

Our experimental subjects were 10 undergraduate students at the University of California Riverside. They were given a five-minute introduction to both timeboxes and VTTS, and allowed to "play" with both for ten minutes before the experiment began. We measured the quality of the separation Q, achieved by both tools as:

$$Q = 2 * (\textit{Correctly Separated} - \textit{False Positives}) / \textit{Size of Dataset}$$

Because our datasets has equal numbers of each of the two classes 10, this measure is in the range [-1, 1], with 1 indicating perfect separation of the classes. Table 1 summarizes the results of our experiments.

Table 1. The quality of the separation (Q) the two query mechanisms under consideration, on the Cylinder-Bell-Funnel and Control Chart datasets

	Timeboxes	Variable Time Timeboxes
Cylinder-Bell-Funnel	0.27	0.82
Control Chart	0.38	0.78

It is clear from these results that VTTs are a more powerful and intuitive tool for querying time series databases.

4 Related Work

Time series data accounts for an increasingly large fraction of the world's supply of data. A random sample of 4,000 graphics from 15 of the world's newspapers published from 1974 to 1989 found that more than 75% of all graphics were time series [19]. Visualizations of time-series data attempt to improve the utility of these common graphs, through the use of techniques such as increased data density or polar-coordinate displays that emphasize the serial periodic nature of the data set [7], or by distorting the time axis to realize denser information displays [14]. A recent survey of linear temporal visualizations is found in [17]. Generally, these tools focus on visualization and navigation, with relatively little emphasis on querying data sets.

A few tools have been developed for querying time-series data. MIMSY [15] provided an early example of searches for temporal patterns in stock market data, using text entry fields, pull-down menus, and other traditional widgets to specify temporal constraints. QuerySketch is an innovative query-by-example tool that uses an easily drawn sketch of a time-series profile to retrieve similar profiles, with similarity defined by Euclidean distance [20]. Although the simplicity of the sketch interface is appealing, the use of Euclidean distance as a metric can lead to non-intuitive results [11].

Spotfire's Array Explorer [18] supports graphically editable queries of temporal patterns, but the result set is generated by complex metrics in a multidimensional space. This potent approach produces useful results, but users may wish to constrain result sets more precisely.

Support for progressive refining of queries was addressed by Keogh and Pazzani, who suggested the use of relevance feedback for results of queries over time series data [11].

5 Conclusions

The additional expressive power provided by VTTs presents some additional challenges that merit further study. As VTTs require specification of an additional parameter, creation and manipulation will likely be more difficult than is the case with simple timeboxes. Identification of appropriate mechanisms for these tasks, perhaps including evaluation of alternative designs, will be needed to identify a preferred strategy. VTTs also raise questions of semantics: for example, what is the interpretation of overlapping VTTs? The interpretation of overlapping timeboxes is straightforward, but overlapping VTTs might confuse users. Other analogous extensions to the timebox model might also be possible. For example, variable value timeboxes (VVTs) might support variations in values similar to the valuations in time supported by VTTs. These queries would present further challenges in creation, interpretation, and efficient processing.

Acknowledgements. Harry Hochheiser is supported by a fellowship from America Online.

References

1. R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. The *VLDB Journal*, pages 490-501, 1995.
2. R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zait. Querying shapes of histories. *Proceedings of the 21st International Conference on Very Large Databases*, pages 502-514, 1995.
3. R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *Proceedings 11th International Conference on Data Engineering*, ICDE, pages 3-14, Taipei Tawian, March 1995. IEEE Press.

4. B. Bederson, J. Meyer, and L. Good. Jazz: An extensible zoomable user interface graphics toolkit in java, November 2000.
5. D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229-248. AAAI Press/MIT Press, 1996.
6. J. V. Carlis and J. A. Konstan. Interactive visualization of serial periodic data. *ACM Symposium on User Interface Software and Technology*, pages 29-38, San Francisco CA, November 1998. ACM Press.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf Computational Geometry: Algorithms and Applications. Spring-Verlag, 2000.
8. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data*, pages 419-429, Minneapolis Minnesota, May 1994.
9. H. Hochheiser and B. Shneiderman. (2001). Interactive exploration of time-series data, In *Proc. Discovery Science 4th International Conference 2001*, Editors (Jantke, K. P. and Shinohara, A.), Springer-Verlag, Berlin. pp. 441-446.
10. A. Inselberg. Multidimensional detective. *IEEE Conference on Information Visualization*, Phoenix AZ, October 1997.
11. E. J. Keogh and M. J. Pazzani. Relevance feedback retrieval of time series data. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR '99*, pages 183-190, Berkeley CA, August 1999. ACM.
12. J. Little and L. Rhodes. Understanding Wall Street. Liberty Publishing, Inc., Cockeysville MD, 1978.
13. A. Nanopoulos and Y. Manolopoulos. Indexing Time-series Databases for Inverse Queries, *Proceedings 9th International Conference on Database and Expert Systems Applications* (DEXA 98), pages.551-560, Vienna, 1998
14. S. Powsner and E. Tufte. Graphical summary of patient status. The Lancet, 344:386-389, 1994.
15. G. Roth. MIMSY: A system for analyzing time series data in the stock market domain. Master's thesis, University of Wisconsin, Department of Computer Science, 1993.
16. B. Shneiderman. Dynamic queries for visual information seeking. IEEE Software, 11(6):70-77, November 1994.
17. Silva and T. Catarci. Visualization of linear time-oriented data: a survey. *Proceedings of the first International Conference on Web Information Systems Engineering*, Hong Kong, June 2000. IEEE Computer Society.
18. Spotfire. http://www.spotfire.com.
19. E. Tufte. The visual display of quantitative information. Graphics Press, Cheshire, Connecticut. (1983).
20. M. Wattenberg. Sketching a graph to query a time series database. *Proceedings of the 2001 Conference Human Factors in Computing Systems*, Extended Abstracts, pages 381-382, Seattle WA, March 31- April 5, 2001. ACM Press.
21. UCI Knowledge Discovery in Databases Archive: *http://kdd.ics.uci.edu/*

Spatio-Temporal Querying in Video Databases

Mesru Köprülü[1], Nihan Kesim Çiçekli[1,2], and Adnan Yazıcı[1]

[1] Department of Computer Engineering, Middle East Technical University, Ankara, Turkey
mesruk@superonline.com, yazici@ceng.metu.edu/tr
http://www.ceng.metu.edu.tr/~yazici

[2] Department of Computer Science, University of Central Florida, Orlando, USA
nihan@cs.ucf.edu
http://www.cs.ucf.edu/~nihan

Abstract. In this paper a video data model that allows efficient and effective representation and querying of spatio-temporal properties of objects is presented. The data model is focused on the semantic content of video streams. Objects, events, activities performed by objects are main interests of the model. The model supports fuzzy spatial queries including querying spatial relationships between objects and querying the trajectories of objects. The model is flexible enough to define new spatial relationship types between objects without changing the basic data model. A prototype of the proposed model has been implemented. The prototype allows various spatio-temporal queries along with the fuzzy ones and it is prone to implement compound queries without major changes in the data model.

1 Introduction

The uninterruptible advances in computer technology, which have made the storage and processing capabilities increase while the costs decrease, brought multimedia to our computers. As a result, high technology multimedia systems like video on demand services, geographical information systems based on image and video data, digital video libraries, teleconferences, security systems with video support, etc., have become ordinary parts of our life. With the high usage of multimedia systems in our daily life, some new requirements have arisen, like querying the multimedia data. Content-based queries on images have been studied for a relatively long time and many solutions have been developed for storing and querying images in databases. Querying video databases however is a newer concept than image databases.

Although querying video databases is a newer concept, there have been several studies on querying the content of videos [1,8,11,12,13,15,16,17,18,19,20]. There exist mainly two different approaches for content-based query processing in video systems. One approach is to use image-processing techniques in evaluating the queries. The second approach deals with the semantic content of the video data, where the semantic data is stored as metadata with the video. Similar to querying image databases, querying video databases involves queries for spatial relationships among objects. In addition, we need to deal with temporal and spatio-temporal queries in video databases.

Several studies have been done on developing a video model that allows spatial, temporal and spatio-temporal queries. Some of them use annotation-based modelling [7,17], some use physical level video segmentation approach [2,10,19,20], and some have developed object based modelling approaches [1,6,8,13,15,18]. Annotation-based video models allow free text or attribute/keyword annotations in order to describe semantic and spatial attributes of video data. The physical level video segmentation approach does not address semantic concepts, such as, objects, events, roles, players, etc. Instead the video data is described as a stream of small segments with application specific temporal and spatial properties. Object based models focus on the modelling of semantic content of the video data using object-oriented modelling techniques.

Among the object-based models, the Advanced Video Information System (AVIS) [1] introduces indexing video data, based on the objects of interest and events in the video. This index is represented with an association map and then transformed into an augmented interval tree (frame segment tree) for efficient interval access. The main focus in the AVIS is the existence of objects and activities in the video. Objects and activities are stored in special index structures. The model is capable of performing complex and compound content-based queries within the well-defined index structure. However it lacks spatial information and thus, it does not support spatial and spatio-temporal queries.

Since multimedia data is a rich information source, it contains various data types, which are either structured or unstructured. Especially for unstructured data types, handling uncertainty (and fuzziness) is an important issue. Fuzzy queries on multimedia data have already been studied by some researchers [5, 9]. Among these, Aygun and Yazici [4,5], use a fuzzy object oriented conceptual data model [21] to represent fuzzy information and object-oriented concepts [22] of video data.

In this study we present a spatio-temporal data model for video data. The data model is developed as an extension to the AVIS model [1]. We have chosen to use AVIS as the basis for our study, since AVIS is a flexible model that can be used for any kind of video data. It is application-independent, and allows the user to store as much semantic meta-data as required. Our extended data model has the following properties and contributions: (1) it allows to model semantic content of video data including the spatial properties of objects. The extensions to AVIS include extending the association map, so that it can represent spatial properties of objects, being able to store spatial data together with objects in the frame segment tree, and introducing an array structure for objects to store spatial data associated with them, (2) the model allows to perform spatio-temporal queries on the video, including querying spatial relationships between objects in the video and querying moving objects in the video, (3) the model also allows to include fuzziness in spatial and spatio-temporal queries; therefore, one is able to do fuzzy querying, and (4) a prototype implementation of the proposed model is developed. The prototype implementation includes, in addition to all basic queries supported by AVIS, spatial relationship queries, fuzzy spatio-temporal and spatial relationship queries.

The rest of the paper is organized as follows: Section 2 describes the data model in AVIS in detail. In Section 3, we introduce the primary extensions to AVIS, which can handle spatial and temporal properties and relationships between objects. Section 4 discusses the query types that are supported by our model. The last section concludes the paper with some remarks for future extensions.

2 AVIS – Advanced Video Information System

AVIS (Advanced Video Information System) [1], is an object-based video data model that can be used for any kind of video data. In the model, the main focus is on objects, events, and activities, called *entities*, appearing in the video. Objects are the individual entities that may be characters in a movie such as Rambo, James Bond, or they may be objects such as, car, tree, and ball. An activity type may be accepted as the subject of a given frame sequence. It is the type of the action performed in a video (e.g. walking, eating, or kicking a ball). An event is an instance of an activity type. It consists of an activity type, roles in the activity, and objects as the actors of roles in the activity. For example, in the event "Philip is eating a cake", the activity type is eating, the role is eater, and the actor is Philip.

A video stream is composed of a huge set of video frames. A frame sequence is defined as a set of contiguous frames containing semantically important data, like an object or an event. Associated with each entity is a set of frame sequences. These are

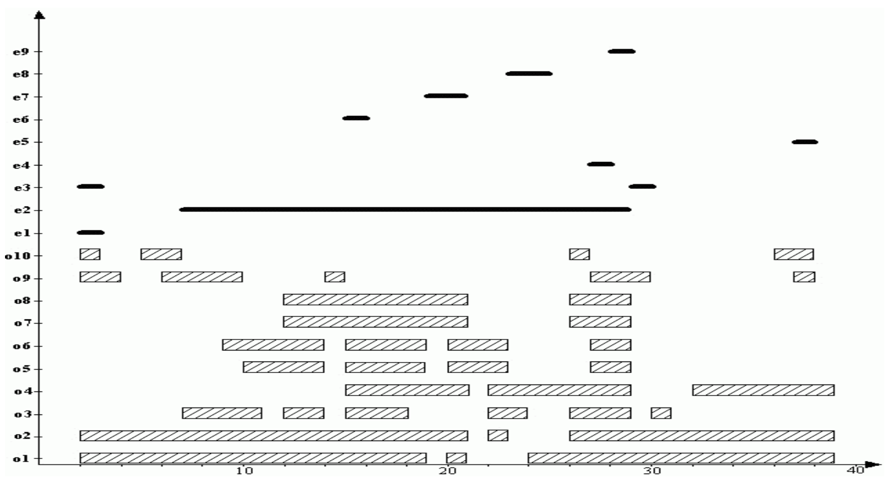


Fig. 1. A sample association map

the frames in which that entity appears in the video. The information about entities and frames is indexed with special index structures based on an association map. Fig. 1 illustrates an association map generated for a movie. The x-axis represents the frames of video and y-axis represents the set of entities in the movie. Only two entity types are presented in the figure, *objects* and *events*. For each entity, the corresponding line segments represent the frames in which they appear. For example, the two line segments of object 7 (*o7*) means that, object 7 appears in frame segments [12…21) and [26…29).

Indexing in AVIS is based on the *frame-segment tree* (FST), which is derived from the association-map. Overlapping line segments for different entities are considered as separate nodes in the frame segment tree. A list of entities that appears in the time interval represented by a node is stored along with the node in the tree. Fig. 2

illustrates the segment tree generated from the map in Fig. 1 for the interval [1-11]. The set of numbers placed in a node denotes the identities of video objects and events

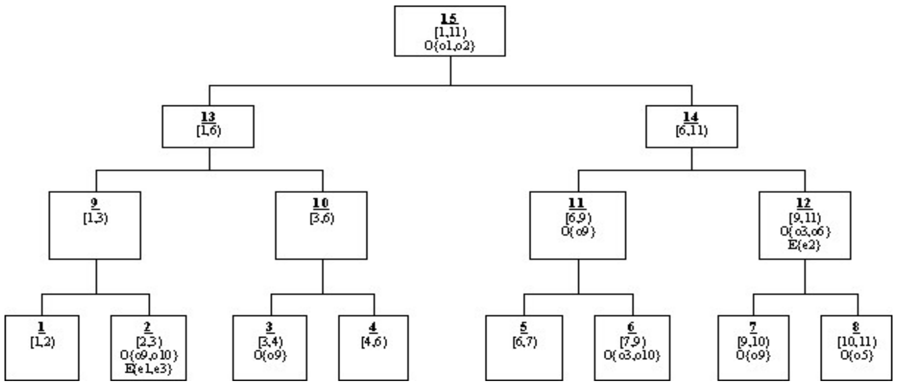


Fig. 2. Frame Segment Tree

that appeared in the entire frame-sequence associated with that node. If an object exceeds the frame interval of a node, then the object is represented in more than one node. For any node, if an object does not appear in all frames of that node, then, the object is inserted into the lists of sub-nodes of the node. For example, the appearance of object *o9* exceeds the time interval represented by node 2. Therefore *o9* is listed in the object list of both node 2 and node 3. (i.e. the intervals [2,3) and [3, 4)). However *o9* cannot be included in the list of node 13, because node 13 represents the interval [1,6) and *o9* does not appear in all frames of that interval.

In addition to the frame segment tree there are some additional index structures to access the nodes of the FST for a given entity. For instance for a given object, it is possible to traverse these index structures to locate the relevant nodes in the frame segment tree, thus to locate the frame sequences in which that object appears.

3 Modeling Spatio-Temporal Properties and Relationships

Handling spatial relationships in video databases is more difficult than in image databases, since video data has time-dependent properties. Some examples of spatial queries that can be applied to video are as follows:

- Find all frames, where first lady stands *close to* President Bush.
- Find all frames, where Bulent Ecevit is *at the right of* George Bush, and Tony Blair is *at the left*.
- Find the name of the Formula1 pilot that is *behind* Michael Schumacher between frames 100 and 120.

In this paper we introduce some extensions to AVIS in order to handle such kind of spatial queries. The extended data model, called spatio-temporal AVIS (ST-AVIS), handles spatial properties of objects in a video and allows the user to express spatial and spatio-temporal queries. In the rest of this section, we first discuss the representation of spatial properties of objects and how different spatial relationships between objects are computed and then show how we incorporate spatial querying into AVIS.

3.1 Spatial Properties of Video Objects

Spatial properties of objects in a video can be represented in different ways. In our model we use 2-dimensional coordinate system, in which the spatial property of the object is its position as appears on the screen. A common strategy is to use approximations for describing an object spatially. The most efficient method for the two-dimensional coordinate system is to use MBRs (Minimum Bounding Rectangles). We make use of a minimum rectangle that covers all parts of an object. This approach is similar to the MBR approach in approximating the location of objects.

The granularity of the coordinate system is an important issue, since a minimum of pixel-level granularity makes it hard to apply to video data. In our model, the user-specified rectangular areas on the screen represent spatial properties. We define the spatial property of an object as follows:

Definition 1. The spatial property of an object A is a *tuple* (R, I), where, R is a rectangular area (a region) that covers all area in which object A appears during the time interval $I=[t_i, t_f]$. R is not exactly the minimum-bounding rectangle of A. At any time t in $[t_i, t_f]$, object A may be located anywhere in R.

Object A may be either static or moving in the rectangle R during the interval I. When object A is spatially static in R, then the rectangle R is practically the minimum-bounding rectangle of A. When A is moving in R, then we assume that the rectangle R is chosen so that A has a uniform movement in R, i.e. in a reasonable distance and as a single moving character. For example, while focusing on an object in the frame, if camera zooms out the object while it is moving, or if the object itself goes away, the size of the rectangle that covers the object gets smaller. In this case, for finer results, the scenes of zoom out should be handled in separate rectangles.

3.2 Spatial Relationships

Since the spatial property of an object is the minimum region that contains the object, the spatial relationship between two objects is obtained from the spatial relationship between the regions of each object. This property gives one the ability to find spatial relationships between any two objects.

Li et. al. [16] have a formal definition of each spatial relation in a rule base. They extended Allen's temporal interval algebra [3] into two-dimensional space in order to define spatial relationships between rectangular areas. They divide spatial relations into two groups; *directional* and *topological*. Directional relations include *south, north, west, east, northwest, northeast, southwest,* and *southeast*. Topological relations include *equal, inside, contain, cover, covered by, overlap, touch,* and *disjoint*. We use the notation of [16] in our formalization of spatial relations. In our notation, we use *left, right, top,* and *bottom* instead of *west, east, north*, and *south*, respectively. Our rule base covers the relations *top, bottom, right, left, top-right, top-left, bottom-right, bottom-left, overlaps, equal, inside, contain, touch*, and *disjoint*. Any two rectangular areas satisfying the definition of any of these spatial relations have those spatial relationship(s) between themselves.

Such a rule base can help the calculation of spatial relationships between two static objects, like in images, accurately. However in videos, objects are not static. They may be moving or they may seem to be moving because of a moving camera or zooming in or out. Therefore we need to deal with spatial relationships that change over time. At a certain snapshot of the video, one object may appear to be exactly to the left of another object. But within an interval of time, being its left might not be that exact all the time. Therefore we introduce fuzziness to the definition of spatial relationships between moving objects in videos. We define this fuzzy spatio-temporal relationship as follows:

Definition 2. Let A_i and A_j be two objects. Their *fuzzy spatio-temporal relationship* during time interval I_k is $A_i(\alpha,\mu,I_k)A_j$ where α is one of the spatial relations supported by our model (top, left, etc.), and μ is the value of membership. $A_i\alpha\mu A_j$ is true during the interval I_k.

Notice that this definition is similar to the definition of spatial relations between two moving objects given by Li et. al [16]. Their definition includes both directional and topological relations. Two different operators are used representing each relation type. We have used only one operator α for both directional and topological relations. However we use a fuzzy membership value, μ which is in [0,1], for any spatial relationship. The membership value determines how much the spatial position between two objects satisfies a given spatial relation.

We present some samples for LEFT relation between two objects in Fig. 3. In this figure, two objects, namely A and B, are assumed. Each occurrence of object A in the figure has a LEFT relationship with object B with different membership values. For a given time interval I and a membership μ, this relationship will be specified by A

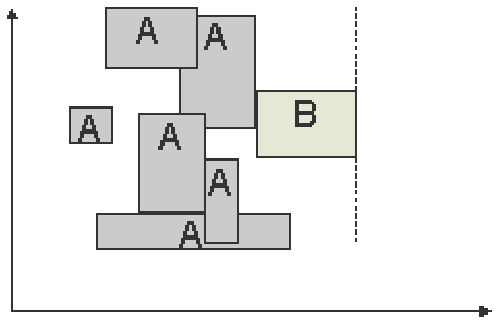


Fig. 3. Examples for LEFT relationship for two objects

Table 1. Relation between the membership value and angle between the centers of rectangles

Relation	**Angle** (*)	**Membership Value**
TOP	arctan(x/y)	1- (angle/90)
LEFT	arctan(y/x)	angle/90
TOP-LEFT	arctan(x/y)	1 – (abs(angle-45) / 45)
TOP-RIGHT	arctan(y/x)	1 – ((angle – 45) / 45)
* x is the horizontal distance between centers of two rectangles. * y is the vertical distance between centers of two rectangles.		

(LEFT, μ, I) B.

The membership value is used in the second step for the calculation of a relationship. The first step is to determine if any two objects satisfy the conditions stated in the rule base. If these conditions are not satisfied, membership value μ is assumed to be 0. Otherwise, we calculate the membership value of the relationship using the centres of rectangles. The angle between the x-axis and the line passing through the centres of the rectangles are used to calculate the membership value. Table 1 illustrates the relation between the angle and the membership value for each relation. Notice that, in this table, we represent only relations, *top, left, top-left*, and *top-right*. The reason is that, these relations are inverses of *bottom, right, bottom-right,* and *bottom-left* respectively. That is, TOP(A,B) is equal to BOTTOM(B,A), and so on. Therefore, only these four relations are enough for the calculation. The calculation of membership values allows us to perform fuzzy queries on video data. The user is able to specify the preciseness of the result of any query.

3.3 Extensions to AVIS

In this section we discuss how we incorporate spatial properties of video data into the AVIS data model. We have extended the contents of the association map and therefore the frame segment tree. Additional indexes to the frame segment tree are also modified.

In AVIS the association map shows the frame sequences in which the entities appear. However an object can appear in different locations in a single frame sequence. Therefore, we need to divide each frame sequence into sub-intervals according to the locations of the object. As a result, we represent *(interval, region)* pairs for objects in the extended association map. Fig. 4 presents a sample drawing of an extended association map where we used different patterns to indicate different locations of an object within a frame sequence. For example, in the figure, the boy appears in two separate frame sequences, but he is seen in three different locations in the first frame sequence and five different locations in second frame sequence.

In AVIS, each node in the frame segment tree has a list of objects existing in the interval represented by this node. In order to implement the region concept, the node structure in the frame segment tree is modified. In our model, a tree node is defined to contain a list of *(object, region)* pairs. This means that, each (o_i, r_i) pair in the list states that, “object o_i appears in the rectangular area r_i within the time interval represented by that node”.

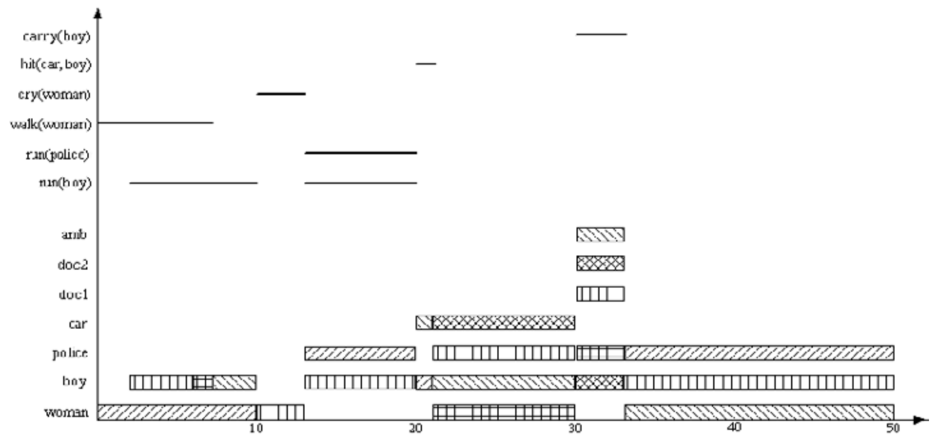


Fig. 4. The extended association map

Storing objects with their regions within the time intervals had an important effect on the frame segment tree. An object *x* may appear in more than one region during a set of continuous frames. For example, a plane crossing the screen from left to right, visits more than one region. In the segment tree of AVIS, this entity is represented in only one FST-node, since the frames are continuous. However, in spatio-temporal model (ST-AVIS), we divide this single interval into sub-intervals, when the region of the object changes. This means that, each of these sub-intervals is represented with a separate tree node. As a result, the number of nodes in the tree increases. This affects the cost of operations on the tree proportional to the order of *log n*, where *n* is the number of nodes in the tree.

In order to achieve fast access to the nodes of the extended frame segment tree for a given object, we also modified the auxiliary indexes for objects. Each element in the object index points to a list of *(interval, region)* pairs, showing the intervals during which the object appears in the stated region. Additionally, for each *(interval, region)* pair, there is a list of pointers each of which points to the nodes in the frame segment tree, which is covered by the interval.

4 Supported Query Types

The model introduced in this paper allows all types of queries that are listed in [1]. In addition to these query types, our model supports querying the spatial properties of objects and the spatio-temporal relations between objects as well. We list the basic spatial queries into three categories:

- *Regional Queries*: Given an object and time interval, one may query the regions it appears. Or, given an object and its region one may query the frames.

- *(Fuzzy) Spatial Relationship Queries:* Given two objects and the spatial relationship between them we can query the frames in the video. Here we use fuzziness in the definition of the spatial relationship.
- *(Fuzzy) Spatio-temporal queries* (also called *Trajectory Queries*): Given an object we may query its trajectories.

The supported spatial query types are discussed in the following along with examples.

4.1 Regional Queries and Fuzzy Spatial Relationship Queries

Given object and interval, find regions: This is a rather simple query to handle with the data model of this study. An example query for this type can be "Give a list of regions, where the ball appears in, between 5th and 10th minutes". While processing the query, we first locate the object in the object index. Then, we trace the *(interval, region)* list of the object and retrieve regions of all intervals that intersect with the given interval in the query. A list of regions is returned as a result. The regions are represented with their *x-y* coordinates.

Given object and region, find frames: An example to this type of query is "List all frame sequences during which a plane appears in a given region". The computation is very similar to the previous one, since regions and intervals are kept in the same structure: The object is located in the index, and then all items in *(interval, region)* list are visited to find the given region. When the region is found, the corresponding interval is added to the resulting list. The intervals are sorted and consecutive ones are merged to obtain a solid list of intervals. When searching for the region we apply fuzziness since the region given in the query is only an approximation of the exact location of the object. Thus the equality of the regions is checked within a threshold.

Given objects and a spatial relation in between, find frames: This type of queries are of the form "Find the set of time intervals, in which object A is seen at left of object B, with a threshold value of 0.7". This is a fuzzy spatial relationship query. The model allows us to calculate this query with a cost proportional to the number of occurrences of the objects. We trace the regions of A, and for each interval, we look at the corresponding tree node, to find the object B in a suitable region. If B occurs and if its region satisfies the relationship within the threshold value, then the interval of current node is returned. As the result of the query, we have a solid set of frame intervals in which the given relationship occurs at least 70%.

4.2 Trajectory Queries

Trajectory is the route of an object on the video in consecutive frames. In a trajectory query, we search both the time interval and the location of an object. A trajectory is composed of a set of sorted *(interval, region)* pairs, such that, any interval except the first one is the successor of the previous one in time, and any region except the first one is a neighbour of its predecessor. A formal definition of trajectory is given as the following:

Definition 3. A trajectory of an object o is a set $T_o = \{(I_n, R_n) \mid n > 1\}$ such that;

i. I_n is the time interval $[t_{ns}, t_{nf}]$,
ii. R_n is the rectangular area defined for object o in time interval I_n,
iii. For any i, where $1 < i \leq n$,
- $t_{(i+1)s} = t_{if} + 1$,
- R_{i+1} is a neighbour of R_i, where neighbourhood is defined in Definition 4.

Definition 4. Given two rectangles R_1 and R_2, with coordinates $(r_1x_1, r_1x_2, r_1y_1, r_1y_2)$ and $(r_2x_1, r_2x_2, r_2y_1, r_2y_2)$ respectively, R_1 is a neighbour of R_2 if and only if one of the following conditions is satisfied:

i. $r_1x_1 = r_2x_2 \wedge [r_1y_1, r_1y_2]$ and $[r_2y_1, r_2y_2]$ *overlaps*
ii. $r_1x_2 = r_2x_1 \wedge [r_1y_1, r_1y_2]$ and $[r_2y_1, r_2y_2]$ *overlaps*
iii. $r_1y_1 = r_2y_2 \wedge [r_1x_1, r_1x_2]$ and $[r_2x_1, r_2x_2]$ *overlaps*
iv. $r_1y_2 = r_2y_1 \wedge [r_1x_1, r_1x_2]$ and $[r_2x_1, r_2x_2]$ *overlaps*

where, *overlaps* is taken from Allen's temporal algebra [3]. Since this algebra can be applied to any one-dimensional space, we used it for x-axis and y-axis.

As an example of querying trajectories, consider a query of the form: "Find the trajectory of an object going from the left to the right of screen", where left and right of the screen are specified in the query interface (typically with the mouse). Here the left of screen and the right of screen are not precisely defined regions. The spatial property definitions of the object may not exactly fit in these stated regions. Therefore we use uncertainty in processing this kind of queries. We match the actual object regions with the imprecise regions given in the query by using approximation techniques. A degree of preciseness is requested with each query, in order to match the starting and the ending locations of the trajectory.

The neighbourhood of two rectangles is another point of uncertainty. We implemented a constant uncertainty degree, which is proportional to the areas of two rectangles. This degree may be user-specified or a commonly accepted value.

In our data model it is possible to find trajectories that start at any region on the screen and end at any region on the screen. Our model allows us to find the trajectories of objects that pass the screen from left to right, or diagonally, or from top to bottom, or in the reverse directions. The returned trajectories may not only be linear trajectories but also non-linear, even spiral.

5 Conclusions and Future Work

In this paper we presented a spatio-temporal video data model, which is an extension of the model named AVIS [1]. Our model represents the semantic content of video streams and allows users to model any kind of objects, events, and activities of interest in the video. The model presented here identifies spatial properties of objects with rectangular areas (regions) resembling MBRs. It is possible to compute spatial relationships between two rectangular areas, hence the objects covered by those rectangles. We can handle spatial relations left, right, top, bottom, top-left, top-right, bottom-left, bottom-right, as directional relations, and overlaps, equal, inside, contain, touch, and disjoint as topological relations. We also allow querying the trajectory of an object given the start and stop regions for the required trajectory.

A prototype of the proposed model is developed and implemented [14]. The implementation covers a data entry interface that allows users to do detailed frame-by-frame investigation on the video stream. It is also possible to see the current association map of any video stream graphically within the implementation. The application stores data for more than one video stream in its database. In case of a query development, there is a query interface to generate spatial and non-spatial queries on the video.

One advantage of this study is that, the proposed model is flexible in both design and implementation, so that it is easy to extend it with new spatial relations. New query types on object trajectories can be developed without much effort.

As a future study, the prototype can be improved to be a complete video database system. The other topological relationships such as *near, touch, cover*, can be defined without changing the proposed data model. We did not discuss the implementation of compound and conjunctive queries. However, we think that we have developed a base for these query types, although a further work is necessary for more efficient algorithms.

References

1. Adali, S., Candan, K.S., Chen, S., Erol, K., Subrahmanian, V.S.: The Advanced Video Information System: data structures and query processing, Multimedia Systems, vol. 4, 1996, pp. 172-186.
2. Ahanger, G., Little, T.D.C.: Data Semantics for Improving Retrieval Performance of Digital News Video Systems, IEEE Transactions on Knowledge and Data Engineering, Vol.13 No.3, pp. 352-360, June 2001.
3. Allen, J.F.: Maintaining Knowledge About Temporal Intervals, Comm. ACM, vol.26, pp.832-843, Nov. 1983.
4. Aygün, R.S., Yazıcı, A., Arıca, N.: Conceptual Data Modeling of Multimedia Database Applications, Proc. of IEEE Intl. Workshop on Multimedia DBMS, Ohio, pp. 182-189, August 1998.
5. Aygün, R.S., Yazıcı: A.: Modeling and Management of Fuzzy Information in Multimedia Database Applications, Multimedia Tools and Applications (to appear).
6. Chen, Y.R., Meliksetian, D.S., Chang, M.C.S., Liu, L.J.:Design of a Multimedia Object-Oriented DBMS, Multimedia Systems, vol. 3, pp. 217-227, 1995.
7. Davenport, G., Smith, T.A., Pincever, N.: Cinematic Primitives for Multimedia, IEEE Computer Graphics & Applications, pp. 67-74, July 1991.
8. Dönderler, M.E., Ulusoy, O., Güdükbay, U.: A rule-based video database system architecture, Information Sciences, Vol. 143, No. 1-4, pp. 13-45, June 2002.
9. Fagin, R.: Fuzzy Queries in Multimedia Database Systems, Proc. of 17th ACM SIGACT-SIGMOD-SIGART Symp. on PODS, Seattle, June 1998.
10. Gibbs, S., Breiteneder, C., Tsichritzis, D.:Data Modeling of Time-Based Media, Proc. ACM SIGMOD Conf. On Management of Data, Minnesota, pp. 91-102, June 1994.
11. Hacid, M.S., Decleir, C., Kouloumdijan, J.:A Database Approach for Modeling and Querying Video Data, IEEE Transactions on Knowledge and Data Engineering, Vol.12 No.5, pp. 729-750, October 2000.
12. Hjelsvold, R., Midtstraum, R.: Modeling and Querying Video Data, Proc. Intl. Conf. on Very Large Databases, Santiago, Chile, pp. 686-694, September1994.

13. Jiang, H., Elmagarmid, A.K.: Spatial and Temporal Content-Based Access To Hypervideo Databases, The VLDB Journal, vol. 7, pp. 226-238, 1998.
14. Koprulu, M.:A Spatio-Temporal Video Data Model, MSc Thesis, Middle East Technical University, Department of Computer Engineering, 2001.
15. Kuo, T.C.T., Chen, A.L.P.: A content-based video query language for video databases, Proc. of IEEE Int. Conf. on Multimedia Computing and Systems, Japan, IEEE Computer Society Press, pp. 209-214, June 1996.
16. Li, J.Z., Özsu, M.T., Szafron, D.: Modeling of Moving Objects in a Video Database, Proc. of IEEE Intl. Conf. on Multimedia Computing and Systems, Canada, pp. 336-343, 1997.
17. Little, T.D.C., Ahanger, G., Folz, R.J., Gibbon, J.F., Reeve, F.W., Shelleng, D.H., Venkatesh, D.: A Digital on Demand Video Service Supporting Content-Based Queries, Proc. ACM Multimedia, pp. 427-436, 1993.
18. Oomoto, E., Tanaka, K.: OVID: Design and Implementations of a Video-Object Database System, IEEE TKDE, 5,4, pp. 629-643, August 1993.
19. Swanberg, D., Shu, C.F., Jain, R.: Architecture of a multimedia information system for content-based retrieval, Audio Video Workshop, San Diego, California, 1992.
20. Weiss, R., Duda, A., Gifford, D.: Content-based access to algebraic video, Proc. First IEEE Int. Conf. On Multimedia Computing and Systems, 1994, Boston, MA. IEEE Computer Society Press, pp. 140-153, 1994.
21. Yazıcı, A. et al.: Handling Complex and Uncertain Information in the ExIFO and NF^2 Data Model, IEEE Transactions on Fuzzy Systems, Vol.7, No.6, December 1999.
22. Koyuncu, M., Yazici, A.: IFOOD: An Intelligent Fuzzy Object-Oriented Database Architecture, IEEE Trans. on Knowledge and Data Engineering (to appear).

A Similarity-Based Unification Model for Flexible Querying

S. Krajči[1], R. Lencses[1], J. Medina[2], M. Ojeda-Aciego[2], and P. Vojtáš[3]

[1] Inst. of Informatics. P.J. Šafárik University. Slovakia [†]
[2] Dept. Matemática Aplicada. Univ. Málaga. Spain [‡]
[3] Inst. of Computer Science. Acad. Sci. of the Czech Republic [§]

Abstract. We use the formal model for similarity-based fuzzy unification in multi-adjoint logic programs to provide new tools for flexible querying. Our approach is based on a general framework for logic programming, which gives a formal model of fuzzy logic programming extended by fuzzy similarities and axioms of first-order logic with equality. As a source of similarities we consider different approaches, such as statistical generation of fuzzy similarities, or similarities generated by some information retrieval techniques or similarities arising from fuzzy conceptual lattices.

1 Introduction

There has been increasing interest in the development of formal tools to handle problems of users posing queries and systems producing answers. This focus has become highly relevant as the amount of information available from local and distributed information bases has increased drastically due to the expansion of the world wide web. The recent interest in search engines, and the increasing needs for adding quality in terms of flexibility, performance and precision to such engines, has further added to the importance of the topic of flexible query answering systems, the focus being to add flexibility to the systems for the storage and access to information.

It is customary to consider the following paradigm for flexible query answering: think about an expert human intermediary who is able to analyse users information needs and to evaluate the relevant information items from the available information sources. The knowledge on the information sources and the capability to interpret the user requests enable the expert to perform a good estimate of the items possibly satisfying the users needs, though the query, per se, may be *imprecise*, *incomplete*, or *vague*. Thus, one of the key issues for defining a flexible query answering system is the tolerance to imprecision and uncertainty in the formulation of user queries as well as in the representation of information. A significant effort has been made in representing imprecise information

[†] Partially supported by Slovak grant VEGA 1/7557/20.
[‡] Partially supported by Spanish DGI project BFM2000-1054-C02-02.
[§] Partially supported by COST project 274.

in database models by using fuzzy methods, and several approaches have been proposed for dealing with these problems; for instance, an access structure for similarity-based fuzzy databases is described in [14].

We are convinced that, in order to have a good approach to flexible query answering, one needs to clearly specify both the procedural and declarative parts of our systems. This is usual practice in mathematical logic, where formal models have clearly defined its syntactical part (which deals with proofs) and its semantical part (dealing with truth and/or satisfaction). When applying logic to computer science, mainly in logic programming, a different terminology is used, and we speak about the declarative part of the formal model (corresponding to truth and satisfaction) and the procedural part, more focused on algorithmic aspects of finding proofs (automated deduction).

In this paper we choose the way of including additional information about fuzzy similarities of different objects and using axioms of equality to transfer properties between these objects, our main aim being not to give practical advice on how to detect and handle similarities in practical applications, but to give a formal model for both the declarative and the procedural part of similarity-based fuzzy unification as a tool for flexible query answering.

For illustration purposes we will present a number of problems from the real world, whose solution needs an adequate treatment of similarity:

- Imagine a holidays database with names of touristic cities, depending on the language in which the database has been developed, we can find "London" in English, which in Spanish is "Londres", in Italian "Londra" and even, why not, could have been mistyped as "Lindon". Does a query using "Londres" unify with a database fact about "Londra" or "London"?
- Other problems come from the need of inter-operability, in which a client requires apparently homogeneous access to heterogeneous servers. This causes, for instance, that web users accessing these information sources usually require a multi-step process utilizing the intelligence of the end-user to navigate and to resolve heterogeneity by applying several similarity criteria [6]. There exist proposals of flexible architecture allowing users to specify a wide range of structured queries through a uniform query interface [1].
- Another example from internet: its initial design was made as an initiative for connecting sites with information stored for direct human processing, but the next generation web is aimed at storing machine processable information. For instance, implementing search engines based on ontologies to find pages with words that are syntactically different but semantically similar [4].

The interpretation of our notion of fuzziness is different from probabilistic, possibilistic, believe, ... and we do not discuss their interrelations here. Nevertheless, we can say that on the syntactical (and computational) level our approach is more general. Namely the computational step in possibilistic logic is exactly our fuzzy modus ponens with min connective. This is also the point on the proof that annotation logic is more general than possibilistic, namely that syntacticaly we can in annotation logic mimic computation of possibilistic one (of course restrictic to min, max connectives).

As a source of similarities we consider different approaches, in the next sections, such as statistical generation of fuzzy similarities, or similarities generated by some information retrieval techniques or similarities arising from fuzzy conceptual lattices. Finally, a formal model for similarity-based fuzzy unification is presented by using the multi-adjoint logic programming paradigm, and its relation with Sessa's approach [11] is stressed.

2 Similarity of Documents

Information retrieval is an important branch of computer science which studies the representation and searching of information. Pieces of information are typically stored as a fully (or semi-)structured text in documents. When a user creates a query, an information retrieval system finds documents whose content is relevant to user request. The similarity plays an important role in this process as we will show later. Formally, an information retrieval system can be defined as a triple $I = (D, R, \delta)$, where D is a document collection, R is a set of queries and $\delta \colon R \to 2^D$ is a mapping assigning a set of relevant documents to each query. A widely used document representation is the vector space model (or "bag of words").

Table 1. Vector space representation of a document.

	t_1	t_2	$\dots$	t_n
d_1	w_{11}	w_{12}	$\dots$	w_{1n}
d_2	w_{21}	w_{22}	$\dots$	w_{2n}
$\dots$	$\dots$	$\dots$	$\ddots$	$\dots$
d_m	w_{m1}	w_{m2}	$\dots$	w_{mn}

Each document d_i from a document collection D is represented by a vector $(w_{i1}, \dots, w_{in})$, where w_{ij} is a measure of the importance (weight) of term t_j in document d_i. Terms t_j form a vector of terms (in the table with size n), which contains all the meaningful terms from the whole collection D. Weights can be determined by the well-established method TFIDF of document classification:

$$w_{ij} = \mathit{tf}_{ij} \cdot \log\left(\frac{M}{\mathit{df}_j}\right)$$

where tf_{ij} represents term frequency (TF), that is, the number of occurrences of term t_j in document d_i; and the second factor is the inverse document frequency (IDF) where M is the number of documents and df_j is document frequency, that is, the number of documents in which the term t_j occurs. The TFIDF method gives greater weight to terms which appear more frequently in lesser documents. There are also some variants of this method.

Usual similarity of documents (or documents and query) is based on the euclidean distance or cosine of angle of two vectors:

$$Sim(Q, d_i) = \frac{\sum_{j=1}^{n} w_{ij} \cdot w_{qj}}{\sqrt{\sum_{j=1}^{n} (w_{ij})^2 \cdot \sum_{j=1}^{n} (w_{qj})^2}}$$

where $(w_{q1}, \ldots, w_{qn})$ is the vector of the query Q.

The similarity of documents is typically used in clustering of documents, which can be used for the visualization of a document collection, browsing without querying or quickly finding of documents similar to required one. The similarity of document and the query (also considered as a document) is used in finding documents relevant to the query, which is the primary purpose of information retrieval systems.

3 Similarity on Documents via Fuzzy Conceptual Lattices

There exists another interesting approach to obtain similarity of documents. Let us assume that numbers $R(d_i, t_j) = w_{ij}$ in the matrix from the Table 1 in the interval $[0, 1]$, and take some fix $\alpha \in [0, 1]$.

For every subset N of documents define the set ct_α of all common terms with degree at least α:

$$\mathrm{ct}_\alpha(N) = \{t : (\forall d \in N) R(d, t) \geq \alpha\},$$

and for every subset P of terms define the set cd_α of all common documents with degree at least α:

$$\mathrm{cd}_\alpha(P) = \{d : (\forall t \in P) R(d, t) \geq \alpha\}.$$

An α-concept is defined as a pair (N, P) of subsets of documents and terms, respectively, such that $\mathrm{ct}_\alpha(N) = P$ and $\mathrm{cd}_\alpha(P) = N$.

It is easy to see that set L_α of all α-concepts with ordering $\leq$, defined by $(N_1, P_1) \leq (N_2, P_2)$ iff $N_1 \subseteq N_2$ (or, equivalently, $P_2 \subseteq P_1$), is a lattice, so-called *fuzzy conceptual lattice*.

It is not difficult to define some type of similarity of documents in this context: Documents d_1 and d_2 are said to be α-equivalent iff there is no set of documents N in the lattice L_α which separates them, i.e. for all N, either both d_1 and d_2 belong to N or neither does. The difference of d_1 and d_2 is defined as

$$\mathrm{diff}(d_1, d_2) = \mu(\{\alpha \in [0, 1] : \ d_1 \text{ and } d_2 \text{ are not } \alpha\text{-equivalent}\})$$

where μ is Lebesgue's measure. This difference is a pseudo-metric and, of course, similarity is defined as the complement of this difference to 1.

In practice, values in the table R are usually rational (even decimal) numbers, i.e. there exists some positive integer n that all values have a form ${}^{p(d,t)}/_{n}$ for some integer $p(d,t)$. This fact allows for a simpler definition of the differences function, since for all $p < n$ and all $\alpha \in \left({}^{p}/_{n}, {}^{(p+1)}/_{n}\right]$ all lattices L_α are identical and it follows that

$$\text{diff}(d_1, d_2) = \frac{\text{card}\{p : 1 \leq p \leq n \wedge (d_1 \text{ and } d_2 \text{ are not } \tfrac{p}{n}\text{-equivalent})\}}{n}$$

Note that roles of documents and terms can be interchanged, therefore it can be defined similarity of terms in the same way. Another approach to similarity of terms is given in the next section.

4 Similarity of Terms

To define similarity for two terms we can work either syntactically or semantically. Syntactical approaches to similarity can be based on the definition of distance functions over terms. These functions must be metrics (reflexive, symmetric and satisfy the triangle inequality). One example of such a function is *Hamming distance*, which is defined as the number of positions with different characters in two terms with equal length. Another distance is the *edit* or *Levenshtein* distance. This function is defined as the minimum number of operations (insertion, deletion and substitution of one character) that are necessary to make one term equal to another.

Semantical similarity of terms is part of the relationships between terms included in a typical information retrieval structure, such as a thesaurus. Thesauri can contain equivalences (synonyms and quasi-synonyms), hierarchical structures (hypernyma and hyponyma, meronyma) and associative relationships (other than equivalence or hierarchical relations). Another approach concerns grouping semantically similar terms (synonyms) into clusters.

Thesauri or clusters can be built by hand or automatically generated. The automatic generation of such structures can be based on the joint occurrences of terms in documents. Terms which occur simultaneously very often in documents, should have some relationship (which can be given a name). This idea can be extended with a notion of distance (terms occurring jointly should be in stronger relationship, if they their meaning is closer to each other). Another approach is based on the idea that two terms are similar if their contexts (terms in the neighbourhood) are similar (have relationship)—this is so called indirect similarity.

Similarity of terms is typically used in query expansion. A user constructs the query, then the information retrieval system returns the relevant documents and, finally, the system or the user (or both) can refine the query. Information retrieval systems can analyse retrieved documents (what do they have in common?) and build clusters of terms. These clusters are utilized to expand the query—either the user chooses the proper terms or the system does it automatically. A whole document collection can be used for thesaurus generation.

5 Subjective Similarities

The previous approaches correspond to object-attribute data model, where answering a query we have to fulfil some selection conditions. For instance, in a classical query

```
SELECT  Hotel
FROM    Hotels, Distance, Building
WHERE   Price < 1000, Distance < 50, Age of Building > 1995
```

we have to specify selection conditions using comparison on domains. This approach is not satisfactory because of two reasons:

1. First, it does not rank results starting with the best.
2. If there are too many or no results, we have to iterate the query by changing selection conditions: increasing or decreasing parameters $1000, 50$ and 1995, but we do not know which up which down, in what steps ...

In a fuzzy query we can specify selection conditions by fuzzy sets

```
SELECT  Hotel
FROM   Hotels, Distance, Building
WHERE  Price is Cheap, Distance is Close, Age of Building is New
```

and we can order results by a (user tuned) aggregation, e.g. a weighted sum.

Here we would like to show another advantage of such an approach. Namely, we can deduce similarities on the respective domains from these fuzzy sets.

The approach has a probabilistic motivation: In probability theory there is a procedure which reasonably describes the geometry in the sample space. The likelihood distance is a "natural" pseudometric generated by the distribution function. In [12] a procedure was described for generating such a pseudometric (and hence a similarity) on an arbitrary interval. The only problem is the assumption of differentiability of the distribution.

We can understand our fuzzy sets for "close" and "cheap" as subjective probability density functions (up to some assumption on normalisation). We describe some heuristical construction which is under testing:

– Given a probability measure P with distribution function $F(x) = P((-\infty, x))$ and density $f(x) = dF(x)/dx = \mu$ (where μ is the fuzzy set) an alternative (pseudo)metric describing the geometry of the sample space is defined as
$$\rho(x_1, x_2) = |F(x_1) - F(x_2)|$$
– Another possibility is to avoid derivation and integration and assumptions on this. A similar effect can be obtained by using a monotone function T_μ generated from the density by "inverting" descending parts of the graph of the density (fuzzy set on an ordered domain, e.g. the real line or an interval).

Having a fuzzy set $\mu\colon [a,b] \longrightarrow [0,1]$ take all local maxima $t_1, t_2, \ldots, t_i, \ldots$ and all local minima $b_1, b_2, \ldots, b_i, \ldots$ of the function μ and assume that

$$t_1 < b_1 < t_2 < b_2 < \ldots < t_i < b_i < \ldots$$

The function T_μ is defined by induction through i as follows:

$$\begin{aligned}
x \in (a, t_1] &\text{ then } T_\mu(x) = \mu(x)\\
x \in (t_1, b_1] &\text{ then } T_\mu(x) = T_\mu(t_1) + (\mu(t_1) - \mu(x))\\
x \in (b_1, t_2] &\text{ then } T_\mu(x) = T_\mu(b_1) - (\mu(b_1) - \mu(x))\\
\ldots&\\
x \in (t_i, b_i] &\text{ then } T_\mu(x) = T_\mu(t_i) + (\mu(t_i) - \mu(x))\\
x \in (b_i, t_{i+1}] &\text{ then } T_\mu(x) = T_\mu(b_i) - (\mu(b_i) - \mu(x))
\end{aligned}$$

then the pseudometric $\rho_\mu(x_1, x_2) = |T_\mu(x_1) - T_\mu(x_2)|$ generates a similarity after a normalisation.

6 Multi-adjoint Similarity-Based Unification

We have opted to approach a querying problem (finding an information, or an object of our interest) by using logic methods defined by a suitable declarative and computation model. Here we briefly present a fuzzy similarity-based unification procedure, which is based on a theory of fuzzy logic programming with crisp unification constructed on the multi-adjoint framework introduced in [8].

We recall definitions of declarative and procedural semantics of multi-adjoint logic programming and show our model of similarity-based unification. The fact that this theory of fuzzy unification is developed inside the realm of fuzzy logic programming is very important for later integration of fuzzy similarity-based unification and fuzzy logic programming deduction.

Considering different implication operators, such as Łukasiewicz, Gödel or product implication in the same logic program, naturally leads to the allowance of several adjoint pairs in the lattice of truth-values. This idea is used in to introduce multi-adjoint logic programs, so that it is possible to use a number of different implications in the rules of our programs in a more general set of truth-values (a *multi-adjoint lattice*).

The definition of multi-adjoint logic program is given, as usual in fuzzy logic programming, as a set of weighted rules and facts of a first-order language $\mathfrak{F}$.

Definition 1. *A* multi-adjoint logic program *is a set $\mathbb{P}$ of weighted rules of the form $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ such that:*

1. *The consequent of the implication, A, is an atom which is called the* head.
2. *The antecedent of the implication, $\mathcal{B}$, is called the* body*, and is a formula built from atoms $B_1, \ldots, B_n$ $(n \geq 0)$ by the use of conjunctors, disjunctors, and aggregators.*
3. *The* weight *ϑ is an element (a truth-value) of L.*

Facts *and* goals *or* queries *are understood as usual. Free occurrences of variables in the program are assumed to be universally quantified.*

Note that the weight of a rule is here considered as a degree of relevance of the document, degree of how an answer fits our query. The key point is we that we can use truth functional logic.

Definition 2.

1. *An* interpretation *is a mapping* $I\colon B_{\mathbb{P}} \to L$ *from the Herbrand base of* $\mathbb{P}$ *to the multi-adjoint lattice of truth-values* $\langle L, \preceq\rangle$*.*
2. I satisfies *a weighted rule* $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$*, if and only if its extension to the whole set of formulas* $\hat{I}$ *satisfies* $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$*;*
3. I *is said to be a* model *of a program* $\mathbb{P}$ *if and only if all weighted rules in* $\mathbb{P}$ *are satisfied by* I*.*

It is possible to extend uniquely the mapping I, defined on $B_{\mathbb{P}}$, to be defined on the set of all formulas of the language, this extension is denoted $\hat{I}$. The non-ground case is handled in a straightforward manner, due to the fact that all our formulas are considered universally closed.

Definition 3. *A pair* $(\lambda; \theta)$ *where* $\lambda \in L$ *and* θ *is a substitution, is a* correct answer *for a program* $\mathbb{P}$ *and a query* $?A$ *if for any model of* $\mathbb{P}$ *we have* $\lambda \preceq \hat{I}(A\theta)$*.*

As usual in logic programming, the semantics of a multi-adjoint logic program $\mathbb{P}$ is defined as the least fix-point of the immediate consequences operator $T_{\mathbb{P}}$, which is monotone and continuous (under general hypotheses); as a consequence, the least model can be reached in at most countably many iterations.

The computational model proceeds by substitution of atoms by lower bounds of their truth-value until, eventually, an extended formula with no atom is obtained, which will be interpreted in the multi-adjoint lattice to get the computed answer. Formally, given a program $\mathbb{P}$, we define the following admissible rules:

Definition 4. Admissible rules *for a pair* (F, θ) *where* F *is a formula and* θ *is a substitution, and* A *is an atom occurring in* F *(denoted* $F[A]$*), are the following:*

R1 Substitute $F[A]$ *by* $\big(F[A/\vartheta \mathbin{\bar{\&}_i} \mathcal{B}]\big)\theta'$*, and* θ *by* $\theta' \circ \theta$ *whenever*
  a) θ' *is the mgu of* C *and* A*,*
  b) there exists a rule $\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle$ *in* $\mathbb{P}$*,*

R2 Substitute A *by* $\bot$ *(just to cope with unsuccessful branches), and do not modify* θ*.*

R3 Substitute $F[A]$ *by* $\big(F[A/\vartheta]\big)\theta'$ *and* θ *by* $\theta' \circ \theta$ *whenever*
  a) θ' *is the mgu of* C *and* A
  b) there exists a fact $\langle C \leftarrow_i \top, \vartheta\rangle$ *in* $\mathbb{P}$*.*

Note that if a formula turns out to have no atoms, then can be directly interpreted in the lattice. This justifies the following definition of *computed answer*:

Definition 5. *Let* $\mathbb{P}$ *be a program in a multi-adjoint language interpreted on a multi-adjoint lattice* $\langle L, \preceq\rangle$ *and let* $?A$ *be a goal. An element* $(\dot{@}[r_1, \dots, r_m], \theta)$*, with* $r_j \in L$*, for all* $j = 1, \dots, m$ *is said to be a* computed answer *if there is a sequence* $G_0, \dots, G_{n+1}$ *such that*

1. *$G_0 = (A, id)$ and $G_{n+1} = (\bar{@}[r_1, \dots, r_m], \theta')$ where $\theta = \theta'$ restricted to the variables of A and $r_j \in L$ for all $j = 1, \dots m$.*
2. *Every G_i, for $i = 1, \dots, n$, is a pair of a formula and a substitution.*
3. *Every G_{i+1} is inferred from G_i by one of the admissible rules.*

Assuming the necessary technical hypotheses, the following approximate completeness result was proven in [9]:

Theorem 1 (Approximate-completeness). *Given a program $\mathbb{P}$, for every correct answer $(\lambda; \theta)$ for a program $\mathbb{P}$ and a ground goal $?A$, there is a sequence of computed answers (λ_n, id) such that $\lambda \preceq \sup\{\lambda_n : n \in \mathbb{N}\}$.*

Our approach to similarity-based unification considers similarities acting on elements of domains of attributes. The idea is based on the fact that part of our knowledge base, the multi-adjoint program $\mathbb{P}$, might consist of graded facts representing information about existent similarities on different domains which depend on the predicate they are used. The particular semantics of the multi-adjoint paradigm, enables us to easily implement a version of fuzzy unification by extending suitably our given program.

Given a program $\mathbb{P}$ we construct an extension by adding a parametrized theory E (which introduces a number of similarities depending on the predicate and function symbols in $\mathbb{P}$), such as those below

$$\langle s(x,x), \top\rangle \qquad \langle s(x,y) \leftarrow s(y,x), \top\rangle \qquad \langle s(x,z) \leftarrow s(x,y) \,\&\, s(y,z), \top\rangle$$

For all function symbol we also have

$$\langle s(f(x_1, \dots, x_n), f(y_1, \dots, y_n)) \leftarrow s_1^f(x_1, y_1) \,\&\, \cdots \,\&\, s_n^f(x_n, y_n), \top\rangle$$

Finally, given a predicate symbol, then the following rules are added

$$\langle P(y_1, \dots, y_n) \leftarrow P(x_1, \dots, x_n) \,\&\, s_1^P(x_1, y_1) \,\&\, \cdots \,\&\, s_n^P(x_n, y_n), \top\rangle$$

where & is some conjunction suitably describing the situation formalized by $\mathbb{P}$.

This way we get a multi-adjoint logic program $\mathbb{P}_E$ in which it is possible to get computed answers wrt $\mathbb{P}_E$ with similarity match in unification. This justifies the introduction of a similarity-based computed answer simply as a computed answer with crisp unification on a program extended by axioms of equality.

It is worth to remark the interesting connection existing between this approach and Sessa's weak unification algorithm [11], provided we work with a particular case of the multi-adjoint framework in which $L = [0, 1]$ and the only connectives will be Gödel's implication and Gödel's conjunction.

In [11], similarities are considered to act on constants, function symbols and predicate symbols, so we will assume we have a similarity relation $\mathcal{R}$ acting on $\mathcal{F} \cup \mathcal{P} \cup \mathcal{C}$ (function, predicate and constant symbols) with truth-values ranging in the unit real interval $[0, 1]$. In our approach, the similarity $\mathcal{R}$ is internalized, that is, we include a new predicate symbol in our language, denoted $s_{\mathcal{R}}$.

Now, for every $f, g \in \mathcal{F}$ with $\mathcal{R}(f, g) > 0$ let us extend our logic program by the following schema of axioms

$$\langle s_{\mathcal{R}}(f(x_1, \dots, x_n), g(y_1, \dots, y_n)) \leftarrow_G s_{\mathcal{R}}(x_1, y_1) \,\&_G\, \cdots \,\&_G\, s_{\mathcal{R}}(x_n, y_n), \mathcal{R}(f, g)\rangle$$

which, in the case of constants it is understood as $\langle s_{\mathcal{R}}(c, d), \mathcal{R}(c, d)\rangle$.

In addition, for every $P, Q \in \mathcal{P}$ with $\mathcal{R}(P,Q) > 0$ let us extend our logic program by a schema of axioms

$$\langle P(y_1,\ldots,y_n) \leftarrow_G Q(x_1,\ldots,x_n) \,\&_G\, s_{\mathcal{R}}(x_1,y_1) \,\&_G \cdots \&_G\, s_{\mathcal{R}}(x_n,y_n), \mathcal{R}(P,Q)\rangle$$

With this notation, it is possible to show that Sessa's approach to unification can be embedded in ours.

Theorem 2 ([9]). *Let $P(t_1,\ldots,t_n)$ and $Q(t'_1,\ldots,t'_n)$ be two atoms, assume that some substitution θ is a λ-unifier (for $\lambda \in [0,1]$) obtained by Sessa's weak unification algorithm, then (λ,θ) is a multi-adjoint computed answer to the query $?P(t_1,\ldots,t_n)$ wrt the program $E \cup \mathcal{R} \cup \{\langle Q(t'_1,\ldots,t'_n), 1\rangle\}$.*

It is remarkable that the equality axioms introduced in order to obtain $\mathbb{P}_E$ are simply the similarity-based extension obtained when the 'external' similarity $\mathcal{R}$ is the usual equality relation.

Furthermore Sessa's similarity-based SLD derivation can be completely emulated by our multi-adjoint computation, since it is applied on a classical program, and the only place where uncertainty appears is in the similarity coming from unification. Now, as a consequence of the theorem on emulation of unification by the computational model of multi-adjoint programs, the following theorem holds, in which we are assuming the language of [11] (Defn.7.2).

Theorem 3. *Given a similarity $\mathcal{R}$, a crisp program $\mathbb{P}$ and a goal G_0, and a similarity-based derivation in Sessa's sense*

$$G_0 \Longrightarrow_{C_1,\theta_1,\lambda_1} G_1 \Longrightarrow \cdots \Longrightarrow_{C_m,\theta_m,\lambda_m} G_m$$

the approximation degree of $\theta_1 \cdots \theta_n$ restricted to the variables of G_0 is set to be $\lambda = \min_{1\le i\le m}\{\lambda_i\}$, then there exists a multi-adjoint computation for $?G_0$ and the (crisp) program $\mathbb{P}$ in the logic with Gödel connectives and $L = [0,1]$ such that the computed answer is (λ,θ).

To finish with, it is important to note that all theorems on fix-point, H_λ and $\mathcal{P}_\lambda$ semantics given in [11] state that Sessa's approach perfectly embeds in this more general multi-adjoint approach suitable restricted to the unit interval and Gödel connectives.

7 Conclusions

Different approaches have been introduced to generate similarities to be used in flexible query answering systems, such as statistical generation of fuzzy similarities, or generation by some information retrieval techniques or similarities arising from fuzzy conceptual lattices. We do not mention similarities between fuzzy sets at all, there are several works in the literature on similarity between fuzzy sets [2,3,5,7,10,13]; we use crisp domains of attributes and we use fuzzy sets to generate fuzzy similarities on crisp domains.

Later, a formal model for similarity-based fuzzy unification is presented by using the multi-adjoint logic programming paradigm to provide new tools for flexible querying. The approach gives a formal model of fuzzy logic programming extended by fuzzy similarities and axioms of first-order logic with equality. Finally, it is shown how the given framework perfectly emulates Sessa's approach to similarity-based unification.

References

1. S. Adali and C. Bufi. A flexible architecture for query integration and mapping. In *Cooperative Information Systems Conference*, 1998.
2. B. Bouchon-Meunier, M. Rifqi and S. Bothorel. Towards general measures of comparison of objects. *Fuzzy Sets and Systems*, 84:143–153, 1996.
3. S.M. Chen, M.S. Yeh and P.Y. Hsiao. A comparison of similarity measures of fuzzy values. *Fuzzy Sets and Systems* 72:79–89, 1995
4. S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: the roles of XML and RDF. *IEEE Internet Computing*, 43:2–13, 2000.
5. P. Fonck, J. Fodor and M. Roubens. An application of aggregation procedures to the definition of measures of similarity between fuzzy sets *Fuzzy Sets and Systems* 97:67–74, 1998
6. K.G. Jeffery. What's next in database. *ERCIM News*, 39:24–26, 1999.
7. F. Klawonn and J.L. Castro. Similarity in fuzzy reasoning *Mathware and Soft Computing* 2:336–350, 1995 .
8. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In *Progress in Artificial Intelligence, EPIA'01*, pages 290–297. Lect. Notes in Artificial Intelligence 2258, 2001.
9. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 2002. Submitted.
10. C.P. Pappis and N.I. Karacapilidis. A comparative assessment of measures of similarity of fuzzy values *Fuzzy Sets and Systems* 56:171–174, 1993
11. M.I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275(1–2):389–426, 2002.
12. P. Vojtáš and Z. Fabián. Aggregating similar witnesses for flexible query answering. In H.L. Larsen et al., editor, *Flexible Query Answering Systems, FQAS'00*, pages 220–229. Physica Verlag, 2000.
13. X. Wang, B. De Baets. and E. Kerre. A comparative study of similarity measures *Fuzzy Sets and Systems* 73:259–268, 1995.
14. A. Yazici and D. Cibiceli. An access structure for similarity-based fuzzy databases. *Information Sciences*, 115(1–4):137–163, 1999.

A Formal Model for Data Fusion

Mounia Lalmas

Department of Computer Science, Queen Mary University of London,
London E1 4NS, England, United Kingdom
mounia@dcs.qmul.ac.uk
http://www.dcs.qmul.ac.uk/

Abstract. In information retrieval, the data fusion problem is as follows: given two or more independent retrieved sets of ranked documents in response to the same query, how to merge the sets in order to present the user with the most effective ranking? We propose a formal model for data fusion that is based on the knowledge that can be derived from the retrieved documents. The model is based on evidential reasoning, a theory that formally expresses knowledge and the combination of knowledge. Knowledge characterising a ranked list of retrieved documents is symbolised. The combination of knowledge associated to the several retrieval results yields the characterisation of the merged result.

1 Introduction

The *information retrieval* problem is the quest to find the set of documents relevant to a query. The retrieved documents upon delivery to the user are ranked according to how they have been estimated relevant to the query. A new challenge in information retrieval has arisen with distributed document collections. Some documents may belong to several collections, whereas others belong to a single collection. Also, representation methods and retrieval strategies may vary from collection to collection. Therefore, each collection often produces a different set of documents and a different ranking of documents as a response to a query. The challenge is the following: "how to combine the ranked sets in order to present the user with the most effective ranking?". This is referred in information retrieval as the *data fusion* problem [15,17,1,3,10,12,14,8,16].

This paper proposes a formal model for data fusion based on *evidential reasoning* as developed in [11]. Our approach is based on a formalisation of the *knowledge* that can be derived from the retrieval results. By symbolising the knowledge describing a retrieved set of documents, the *combination of knowledge* yields the knowledge that characterises the merged set of documents. Evidential reasoning formally expresses knowledge and the combination of knowledge.

The paper is organised as follows. First, we provide some background and motivation of our work. Second, we express formally the knowledge describing a retrieval result. Third, we show how the combination of knowledge as defined in evidential reasoning allows the expression of the merged result. Finally, we conclude with some thoughts for future work.

2 Background and Motivation

The data fusion process is divided into three steps: (1) collection selection, which corresponds to the identification of the document collection(s) most likely to contain relevant documents; (2) document selection, which corresponds to determining the number of documents to be retrieved from the selected collection(s); and (3) merging, which corresponds to the actual combination of the individual ranked lists produced by the multiple collections

Collection selection can be implemented by ranking collections, using for example trained queries [17]. This step however increases query time, so the method usually adopted is to treat equally all collections / search engines / strategies (e.g. [15]). The number of selected documents can be determined to be proportional to the quality of each collection [3,17], although most approaches still retrieve an equal number of documents from each collection. Finally, the merging can be based on rank position [17,18], document score [12], or information such as the title or abstract [16]. Similar approaches have been followed by meta-search engines (e.g. ProFusion [6,7], SavvySearch [5], MetaCrawler [13], Fusion [15]).

Various types of information and heuristics are exploited to perform data fusion. It is however not yet possible to determine which types of information or heuristics, or their combination, consistently lead to effective data fusion. This is also because the available information is often not comparable. For instance, many search engines do not return score information, and when they do, these scores are not comparable since they are based on different retrieval strategies as well as statistics upon which these strategies are based.

This paper proposes a formal model for data fusion that can be considered as a meta-model. Our aim is to go beyond the specific approaches that have been developed and implemented, thus obtaining a general framework for representing the data fusion process. The model is based on a formalisation of the knowledge that can be derived from individual ranked lists of retrieved documents. The knowledge describes rank position, score, term appearing in the title and the abstract of a retrieved document, the quality of the a retrieval engine or retrieval strategy, the quality in terms of coverage of the collection itself, etc; the knowledge describes a retrieval result. We can then combine the knowledge describing each retrieval result to arrive at the description of the merged result.

The model proposed in this paper is not specific to a collection or a retrieval strategy. We want a general framework in which we can study the data fusion process. The development of a formal model is not new in information retrieval and has already been shown to lead to the generalisation of information retrieval models. As a result, it was possible to study these models, thus gaining insights about the information retrieval process (see for example [9]). Our ultimate aim is to determine which properties lead to effective data fusion, and which ones do not, thus obtaining a better understanding of the data fusion process. This work is a first step toward this aim, through the development of a formal framework.

3 Representing a Single Retrieval Result

We start by formalising the knowledge that describes a retrieval result. Let a document collection be represented as a set of documents D. From a retrieved set of ranked documents, properties can be derived concerning the documents in D. The derived properties constitute *knowledge* held by an entity, an *agent*, observing the retrieved documents. For instance, from the ranking alone, the agent knows which documents have been estimated more relevant than others. If the titles of the retrieved documents are displayed, the agent may observe (and hence know), for example, that documents containing the term "wine" in their title are ranked higher than those containing the term "water".

In *evidential reasoning*, the knowledge of an agent is formally defined by an *epistemic operator* K. Evidential reasoning as developed by Ruspini [11] provides concepts that formalise the knowledge of the agent. We use these concepts to model the knowledge associated to a retrieval result: the properties describing documents, the documents themselves and the ranking.

3.1 Representing Properties

Knowledge about a retrieval result consists of *known properties* of documents. The properties are formalised upon a *proposition space* and a *sentence space.*

Definition 1 *Let* $P = \{p_0, \ldots, p_n\}$ *be a* proposition space*, where the* p_i*s are propositions.*

The propositions formalises basic properties qualifying a retrieval result. Which properties are effective in describing a retrieval result is an open research question. They also depend on what is available to the agent: ranking alone [15], retrieval status values [2], document titles, parameters associated with the underlying retrieval algorithm, or past retrieval sessions [17]. The properties can include: estimated amplitude of relevance based on some normalisation of the retrieval status values or/and the rank of the retrieved documents [12], effectiveness measure such as precision and recall values (when the underlying retrieval algorithm is known), document content (terms in title, abstract, summary). In this paper, we assume that, for each retrieval result, the properties have been identified and are modelled as propositions of P.

Complex properties of a retrieval result are expressed as objective *sentences* defined upon the proposition space.

Definition 2 *The set of objective sentences is defined as follows: (i) any proposition* p_i *in* P *is an objective sentence; (ii) if* ϕ *and* ψ *are objective sentences, then so are* $\phi \vee \psi$, $\phi \wedge \psi$, $\neg\phi$, *and* $\phi \rightarrow \psi$.

Not all objective sentences (whether they symbolise basic or complex properties of the retrieval set) constitute knowledge. Even if a property is true, it is only when it is known true (by the agent) then its corresponding objective sentence becomes knowledge. This is formally expressed by an *epistemic operator* associated with the agent observing the retrieved result.

Definition 3 *Given an objective sentence ϕ, the sentence* $\mathrm{K}\phi$ *represents that the properties symbolised by ϕ are* known *by the agent. The set of objective sentences and sentences of the form* $\mathrm{K}\phi$ *constitute a* sentence space *denoted S.*

We symbolise the known properties of a document $d \in D$ by a sentence of S: the conjunction of the propositions modelling the known properties of d. For example, let these propositions be p_3, p_4 and $\neg p_8 \vee p_{10}$ for $p_3, p_4, p_8, p_{10} \in P$. Then the sentence $p_3 \wedge p_4 \wedge (\neg p_8 \vee p_{10})$ constitutes knowledge: $\mathrm{K}(p_3 \wedge p_4 \wedge (\neg p_8 \vee p_{10}))$. We denote the function $property : D \longmapsto S$ to assign such a sentence to each document of the collection. In our example, $property(d) \equiv p_3 \wedge p_4 \wedge (\neg p_8 \vee p_{10})$. In the (worst) case when nothing is known about d (for example d has not been retrieved), $property(d) \equiv \bot$, where $\bot$ represents the *false* proposition. The *true* proposition is denoted $\top$.

Basic and complex properties which can characterise a retrieval result (retrieved and non-retrieved documents) have been symbolised, as well as those that are known to characterise the retrieval result. The next step is to formally represent documents.

3.2 Representing Documents

We use *possible worlds* to formalise documents.

Definition 4 *A possible world is an interpretation $w : S \longmapsto \{t, f\}$ that satisfies the axioms of the modal logic system* S5 [4]*, where t and f denote the truth values* true *and* false*, respectively. The set of all possible worlds for S, called the* universe*, is denoted U.*

Given a proposition space P, there can be a maximum of $2^{|P|}$ possible worlds (e.g., one in which all the p_is are true, one in which $p_2, \ldots, p_n$ are true and $\neg p_1$ is true, etc.). This number can be smaller when, for example, two propositions symbolise properties that can never be used jointly to describe a document. For example, if the sentence $p_i \rightarrow p_j$ is true (e.g., $p_i =$"retrieved after rank i" and $j < i$), we cannot have p_i true and p_j false in the same world.

A document $d \in D$ is represented by the set of possible worlds in which the sentence $property(d)$ is true. This is formally defined by a function $world : D \longmapsto U$.

Definition 5 *For $d \in D$, $world(d) = \{w \in U | w(property(d)) = t\}$. Furthermore, the sentence* $\mathrm{K}(property(d))$ *is true in all worlds in $world(d)$.*

Suppose that $P = \{a, b\}$ with four possible worlds given in Table 1 (e.g., $w_1(a) = w_1(b) = t$; $w_2(a) = t$ and $w_2(b) = f$; etc.). Let $D = \{d_1, d_2, d_3\}$ where $property(d_1) \equiv a \wedge b$, $property(d_2) \equiv a$ and $property(d_3) \equiv \bot$ (d_3 has not been

Table 1. Possibles worlds for $P = \{a, b\}$ and $U = \{w_1, \dots, w_4\}$

Worlds in U	True sentences
w_1	a, b,K$(a \wedge b)$,Ka
w_2	$a, \neg b$,Ka
w_3	$\neg a, b$
w_4	$\neg a, \neg b$

retrieved). Then $world(d_1) = \{w_1\}$, $world(d_2) = \{w_1, w_2\}$ and $world(d_3) = \emptyset$ ($\perp$ is true in no world). In addition, K$(a \wedge b)$ and Ka are true in w_1, and w_1 and w_2, respectively. Note that and $\top$ and K$\top$ are true in all worlds.

Additional knowledge about a document can be inferred. This is because some sentences can imply others ($a \wedge b \rightarrow a \vee b$ is true in all worlds). From the axioms of the modal logic system S5, if K$(\phi_1 \rightarrow \phi_2)$ is true (i.e., the agent knows that properties symbolised by ϕ_1 implies properties symbolised by ϕ_2) then so is K$\phi_1 \rightarrow$ Kϕ_2. Hence if K$(a \wedge b)$ is true in w_1 (where $w_1 \in world(d_1)$) then K$(a \vee b)$ is also true in w_1, thus providing additional knowledge about document d_1.

Some of the sentences known true in a world $w \in world(d)$ play a particular role for representing the document $d \in D$.

Definition 6 *The sentence ϕ logically implies the sentence ψ, denoted $\phi \Rightarrow \psi$, iff if ϕ is true at a possible world w, then ψ is also true in that world w.*

Definition 7 *An objective sentence ϕ is said the* most specific sentence *in w if it is known in w and for every objective sentence $\psi \in S$, the sentence* Kψ *is true in w iff $\phi \Rightarrow \psi$.*

For any world $w \in U$, such a sentence always exists because it can always be constructed as the conjunction of all sentences ψ_i such that Kψ_i is true in w (see the proof in [11]). The most specific sentence of a world $w \in world(d)$ corresponds to the most specific knowledge associated to the document d *with respect to that world*. A document can have several most specific sentences.

In our previous example, we have $a \wedge b$ and a as the most specific sentences of w_1 and w_2, respectively, and $w_1, w_2 \in world(d_2)$; hence, d_2 has two most specific sentences, one with respect to w_1, and one with respect to w_2.

The set of most specific sentences of a document is symbolised by the function $mss : D \mapsto \wp(S)$. For $d \in D$, the set of most specific sentences for document d, $mss(d)$, can be derived from $world(d)$. A definition is required first.

Definition 8 *$e(\phi)$ denotes the epistemic set containing all possible worlds that have ϕ as their most specific sentence.*

Some epistemic sets may be empty. In our example, we have $e(b) = \emptyset$.

Theorem 1 *For $d \in D$, $mss(d) = \{\phi \in S | e(\phi) \subseteq world(d)\}$.*

Proof: For $d \in D$ and $\phi \in mss(d)$, there exists $w \in world(d)$ such that ϕ is the most specific sentence in w: $w \in e(\phi)$. For all $w' \in e(\phi)$, all the known sentences of w and w' are those implied by ϕ. Therefore, $w(\mathrm{K}\psi) = w'(\mathrm{K}\psi)$ for all $\psi \in S$ such that $\phi \Rightarrow \psi$. Therefore, $w' \in world(d)$ so we obtain $e(\phi) \subseteq world(d)^1$.

To prove the reverse, we show that for $\phi \in S$, if $e(\phi) \not\subseteq world(d)$ then $\phi \notin mss(d)$. For $\phi \in S$ such that $e(\phi) \not\subseteq world(d)$, there exists a world $w \in e(\phi)$ such that $w \notin world(d)$. For all other worlds $w' \in e(\phi)$, w and w' yield the same truth value for sentences of the form $\mathrm{K}\psi$. If w is not used to represent d, then so is w': $w' \notin world(d)$. Since $e(\phi)$ contains all worlds that have ϕ as their most specific sentence, and none of them belong to $world(d)$, then ϕ cannot be a most specific sentence for d; hence $\phi \notin mss(d)$. □

In our previous example, $mss(d_2) = \{a \wedge b, a\}$. We have also $e(a \wedge b) = \{w_1\}$ and $e(a) = \{w_2\}$. Both $e(a \wedge b) \subseteq world(d_2)$ and $e(a) \subseteq world(d_2)$.

We have modelled the documents of a retrieval result and their known properties. We can also formally reason about the properties. The next step is to represent the ranking itself.

3.3 Representing the Ranking

So far we have used only the logical basis of evidential reasoning. This theory has a second basis, probability theory. A probability distribution is defined upon an algebra of U (see [11] for details), upon which a mass function is constructed to quantify the uncertainty of sentences in S:

Definition 9 *A* mass function *is a mapping* $m : S \mapsto [0, 1]$ *such that:*

$$m(\bot) = 0 \quad \textit{and} \quad \sum_{\phi \in S} m(\phi) = 1$$

The mass function is not a probability distribution, and knowing the details of its construction is not necessary to the understanding of this paper and hence is omitted. What should be retained about m is that for $\phi \in S$:

$$m(\phi) = \begin{cases} > 0 & \text{if } e(\phi) \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Only sentences that are most specific (with respect to some worlds) have a non-null mass value.

We use m to capture the impact of properties in ranking documents. The basic idea is that a sentence describing document d, which *should* be ranked higher than document d', has a higher mass value to that of a sentence describing document d'. Let $\phi, \phi' \in S$ such that $e(\phi) \neq \emptyset$ and $e(\phi') \neq \emptyset$. This is formally expressed as follows:

$$\begin{cases} m(\phi) \geq m(\phi') & \text{if } \phi \text{ qualify documents that should be retrieved (at least)} \\ & \quad \text{before those qualified by } \phi', \\ m(\phi) < m(\phi') & \text{otherwise.} \end{cases}$$

¹ In fact, for any sentence $\phi \in S$, either $e(\phi) \subseteq world(d)$ or $e(\phi) \cap world(d) = \emptyset$.

The estimation of the mass function m requires a study of the document collection and the retrieval set from (whatever) evidence is available to the agent. The worst case is when only ranking information is available. The only properties that can be extrapolated concern the ranks themselves: a proposition p_i expresses the property that a document is ranked ith. Let a document d be ranked ith. We set $property(d) \equiv \phi_i$ where $\phi_i \equiv \neg p_1 \wedge \ldots \wedge \neg p_{i-1} \wedge p_i \wedge \neg p_{i+1} \wedge \ldots \wedge \neg p_l$ where l is the number of retrieved documents; ϕ_i means that the document has been retrieved at rank i and no other rank. If N is the number of documents in the collection, then we have N worlds w_i where $w_i(p_i) = t$ and $w_i(p_j) = f$, for $i \neq j$ (we cannot have p_i and p_j for $i \neq j$ both true in a world). Furthermore, for a document d retrieved at rank i, $world(d) = \{w_i\}$ and ϕ_i is the most specific sentence in w_i. $m(\phi_i)$ can be estimated as follows:

$$m(\phi_i) = \frac{l - (i-1)}{1 + \cdots + l}$$

which leads to the wanted inequality $m(\phi_i) > m(\phi_{i+1})$ since ϕ_i qualifies documents ranked before those qualified by ϕ_{i+1}. $\perp$ characterises the non-retrieved documents and by definition $m(\perp) = 0$.

The rank of a document is expressed in terms of a belief function defined upon the mass function.

Definition 10 *A belief function $Bel : S \mapsto [0,1]$ is defined as follows:*

$$Bel(\phi) = \sum_{\psi \Rightarrow \phi} m(\psi)$$

Definition 11 *The rank of a document is given by the function $R : D \mapsto [0,1]$. For $d \in D$[2], $R(d) = Bel(property(d))$.*

Documents are ranked in decreasing order of the value $R(.)$. If $R(d) = 0$ then document $d \in D$ is not retrieved.

Theorem 2 *For $d \in D$, $R(d) = \sum_{\phi \in mss(d)} m(\phi)$.*

To prove this theorem let us prove first the following lemma.

Lemma 1 *Let $d \in D$. For all $\phi \in mss(d)$, $\phi \Rightarrow property(d)$.*

Proof: Let $\phi \in mss(d)$. Let $w \in U$ such that ϕ is true in w. We have two cases: (1) $w \in e(\phi)$ so from Theorem 1 $w \in world(d)$, and hence $\mathrm{K}(property(d))$ is true in w. (2) $w \notin e(\phi)$, then there exists $\phi' \in mss(d)$ such that $\phi' \Rightarrow \phi$. We have also $e(\phi') \subseteq world(d)$ from Theorem 1, so for the same reason as above, $\mathrm{K}(property(d))$ is true in w. Therefore, for any world where $\mathrm{K}\phi$ is true, $\mathrm{K}(property(d))$ is true, thus $\phi \Rightarrow property(d)$. □

[2] Ranks are usually integer. In our case they are real numbers, but this is not a problem because what is important is to have an ordering.

We prove now Theorem 2.

Proof: Let $d \in D$. From the definition of Bel, $R(d) = Bel(property(d)) = \sum_{\phi \Rightarrow property(d)} m(\phi)$. With Lemma 1, for $\phi \in mss(d), \phi \Rightarrow property(d)$. All sentences in $mss(d)$ are most specific, so $e(\phi) \neq \emptyset$, hence $m(\phi) > 0$. Each such $m(\phi)$ contributes to the value of $Bel(property(d))$:

$$Bel(property(d)) \geq \sum_{\phi \in mss(d)} m(\phi)$$

A most specific sentence $\phi \notin mss(d)$ cannot implies $property(d)$, and hence does not contribute toward the value of $Bel(property(d))$. All none most specific sentences, whether they imply or not $property(d)$ have null mass value. Therefore, we obtain the wanted equality. □

Theorem 2 shows that in our fusion approach, the rank of a document can be based on those most specific sentences that imply the document property sentence, when an appropriate mass function m is constructed to capture the impact of the most specific sentences in ranking documents. The mass function is built upon the knowledge (evidence) observable from a retrieval result. This complete the formal representation of a single retrieval result.

3.4 Working Example

In the remaining of this paper, we will use a working example. Suppose that $D_1 = \{d_1, \ldots, d_4\}$ and $D_2 = \{d_1, \ldots, d_5\}$ (note that d_5 is not a document of collection D_1). We assume the following sets of worlds $U_1 = \{w_1, \ldots, w_6\}$ and $U_2 = \{w'_1, \ldots, w'_5\}$. Let the rankings from collections D_1 and D_2 be referred to as $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. The ranked documents, the worlds representing them, and the set of most specific sentences are shown in Table 2. In the example, for simplicity, each document has a unique most specific sentence. Therefore, for $d \in D_i$, $property_i(d)$ is a most specific sentence and the only most specific sentence in $mss_i(d)$. Note that the documents d_1 and d_4 have not been retrieved in ranking $\mathcal{R}_2$ ($property(d_1) \equiv property(d_4) \equiv \bot$).

Table 2. Ranked documents, the associated worlds and most specific sentences

Ranking $\mathcal{R}_1$			Ranking $\mathcal{R}_2$		
Documents	$world_1(d)$	$mss_1(d)$	Documents	$world_2(d)$	$mss_2(d)$
d_1	$\{w_1, w_2\}$	$\{\phi_1\}$	d_3	$\{w'_1\}$	$\{\phi'_1\}$
d_2	$\{w_1, w_2\}$	$\{\phi_1\}$	d_5	$\{w'_2, w'_3\}$	$\{\phi'_2\}$
d_3	$\{w_3\}$	$\{\phi_2\}$	d_2	$\{w'_2, w'_3\}$	$\{\phi'_2\}$
d_4	$\{w_4\}$	$\{\phi_3\}$			$\top$

The values of the mass functions m_1 and m_2 associated with $\mathcal{R}_1$ and $\mathcal{R}_2$ are given in Table 3. For instance, for ranking $\mathcal{R}_1$, the sentence ϕ_1 characterises

Table 3. Representation of the ranking

Ranking $\mathcal{R}_1$			Ranking $\mathcal{R}_2$		
Sentences	$e_1(.)$	$m_1(.)$	Sentences	$e_2(.)$	$m_2(.)$
ϕ_1	$\{w_1, w_2\}$	0.4	ϕ'_1	$\{w'_1\}$	0.5
ϕ_2	$\{w_3\}$	0.3	ϕ'_2	$\{w'_2, w'_3\}$	0.3
ϕ_3	$\{w_4\}$	0.2	$\top$	$\{w'_4, w'_5\}$	0.2
$\top$	$\{w_5, w_6\}$	0.1			

documents that should be ranked before those characterised by the sentence ϕ_2. From the above assumption (unique most specific sentence) and Theorem 2, for $d \in D_i$, $R_i(d) = Bel_i(property_i(d)) = m_i(property_i(d))$, which as required, produces the rankings shown in Table 2.

We discuss the use of $\top$ for instance for ranking $\mathcal{R}_1$. There is no knowledge with respect to properties covered by worlds w_5 and w_6. Since $\top$ is true in all worlds, then $\top$ is the only knowledge associated to these worlds, and then constitute the most specific sentence for both worlds.

For collection D_i, the knowledge associated to the ranked set of documents $\mathcal{R}_i$ is formalised with the epistemic operator K_i. The set of documents in the merged ranking is $D_1 \cup D_2$. The merging of two rankings can be viewed as a combination of knowledge. The next section describes the fusion of the two rankings.

4 Representing the Merged Retrieval Results

Merging rankings in a model for data fusion based on evidential reasoning is formalised in terms of a *combination of evidence*. The latter is formalised by an epistemic operator K, where the knowledge of the *combined agent* K is defined in terms of the knowledge of the individual agents K_1 and K_2. As for the retrieval result of a single collection, the merged ranking has properties, documents, and a ranking which are determined upon those of K_1 and K_2.

4.1 Representing Properties of the Merged Result

Let P_1 and P_2 be the proposition spaces associated with rankings $\mathcal{R}_1$ and $\mathcal{R}_2$. Let S_1 and S_2 be the respective sentence spaces.

Definition 12 *The sentence space of the combined ranking, denoted $S_\otimes$, is defined by the axioms: (i) if p is proposition of P_i, then it is a sentence of $S_\otimes$; (ii) axioms of logic defining well-formed sentences.*

(i) means that the propositions modelling properties in the merged result are those modelling properties in the individual retrieval results. (ii) defines sentences symbolising complex properties in the merged ranking.

A document in the combined ranking is also characterised by a sentence describing its known properties in the merged ranking. The sentence can be viewed as the *disjunction* of the two sentences describing the document in $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. First, we extend the definition of $property_i$ for $i = 1, 2$.

Definition 13 *Let* $property'_i : D_1 \cup D_2 \longmapsto S_i$ *the extension of* $property_i$*. For* $d \in D_1 \cup D_2$*:*

$$property'_i(d) \equiv \begin{cases} property_i(d) & \text{if } d \in D_i, \\ \bot & \text{otherwise} \end{cases}$$

The sentence $\bot$ characterises a document in D_1 that is not in D_2 and vice versa. The sentence characterising a document in the merged ranking can now be defined.

Definition 14 *Let* $property_\otimes : D_1 \cup D_2 \longmapsto S_\otimes$*. For* $d \in D_1 \cup D_2$*,* $property_\otimes(d) \equiv property'_1(d) \vee property'_2(d)$*.*

In our working example, we have $property_\otimes(d_1) \equiv \phi_1 \vee \bot \equiv \phi_1$ (d_1 is not retrieved in $\mathcal{R}_2$), $property_\otimes(d_2) \equiv \phi_1 \vee \phi'_2$, $property_\otimes(d_3) \equiv \phi_2 \vee \phi'_1$, $property_\otimes(d_4) \equiv \phi_3 \vee \bot \equiv \phi_3$ (d_4 is not retrieved in $\mathcal{R}_2$), and $property_\otimes(d_5) \equiv \bot \vee \phi'_2 \equiv \phi'_2$ (d_5 is not in D_1). For a document $d \in D_1 \cup D_2$ that is retrieved in neither collection, $property_\otimes(d) \equiv \bot$.

4.2 Representing Documents in the Merged Result

Let U_1 and U_2 be the universes associated with rankings $\mathcal{R}_1$ and $\mathcal{R}_2$. Documents in the combined ranking are also represented by a universe.

Definition 15 *The universe of the combined ranking is denoted* $U_\otimes$*. A possible world in* $U_\otimes$ *is a mapping* $w : S_\otimes \longmapsto \{t, f\}$ *that satisfies the following axioms: (i)* w *satisfies the axiom of modal logic* S5*; (ii) if* ϕ *is a sentence of* $S_\otimes$*, then* $\mathrm{K}\phi$ *is true in* w *iff there exists* ϕ_1 *and* ϕ_2 *in* S_1 *and* S_2*, respectively, such that* $\mathrm{K}\phi_i$ *is true in* w *and* $\phi_1 \wedge \phi_2 \Rightarrow \phi$*.*

(ii) means that the knowledge about the combined ranking comes from the conjunction of sentences in S_1 and S_2 modelling properties in the two rankings, and any sentence that can be derived from the conjunction. For a document to be characterised by a sentence ϕ in the combined ranking, the document must be characterised by two sentences ϕ_1 and ϕ_2 in the two individual rankings (the combined agent K must know this fact), respectively, such that $\phi_1 \wedge \phi_2$ logically implies ϕ.

The worlds in $U_\otimes$ are constructed upon the worlds in U_1 and U_2 by way of a *cartesian projection*.

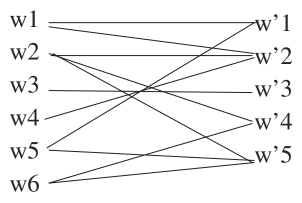


Fig. 1. Compatible worlds between U_1 and U_2

Table 4. Possible worlds in the combined ranking

$U_\otimes$	$\Pi(W_i)$	$\mathrm{K}\phi$	$U_\otimes$	$\Pi(W_i)$	$\mathrm{K}\phi$
W_1	(w_1, w'_1)	ϕ_1, ϕ'_1	W_2	(w_1, w'_2)	ϕ_1, ϕ'_2
W_3	(w_2, w'_2)	ϕ_1, ϕ'_2	W_4	(w_2, w'_4)	ϕ_1
W_5	(w_2, w'_5)	ϕ_1	W_6	(w_3, w'_3)	ϕ_2, ϕ'_2
W_7	(w_4, w'_2)	ϕ_3, ϕ'_2	W_8	(w_5, w'_1)	ϕ'_1
W_9	(w_5, w'_5)	$\top$	W_{10}	(w_6, w'_4)	$\top$
W_{11}	(w_6, w'_5)	$\top$			

Definition 16 *The* cartesian projection *is the mapping $\Pi : U_\otimes \mapsto U_1 \times U_2$ where $\Pi(W) = (w_1, w_2)$ and w_i is the unique possible world in U_i such that for a sentence $\phi \in S_i$: (i) ϕ is true in w_i iff ϕ is true in W; (ii) $\mathrm{K}_i\phi$ is true in w_i iff $\mathrm{K}\phi$ is true in W.*

(i) and (ii) means that the truth assignments to sentences of S_1 and S_2 in $U_\otimes$ worlds must respect those in U_1 worlds and U_2 worlds, respectively. From (ii), the maximum number of $U_\otimes$ worlds that can be created is $|P_1| \times |P_2|$. However, as for the number of possible worlds in the individual case, the number can be smaller since not all U_1 worlds are compatible with all U_2 worlds (e.g., one U_1 world in which p_i is true and one U_2 world in which p_i is false).

For our working example, we assume the compatibility between U_1 worlds and U_2 worlds given in Figure 1. w_1 is compatible with w'_1 and w'_2; w_2 is compatible with w'_2, w'_4 and w'_5; and so forth. The application of definitions 15 and 16 to our working example is shown in Table 4. Take $\Pi(W_1) = (w_1, w'_1)$. $\mathrm{K}_1\phi_1$ and $\mathrm{K}_2\phi'_1$ are true in w_1 and w'_1, respectively, so $\mathrm{K}\phi_1$ and $\mathrm{K}\phi'_1$ are true in W_1. Take now $\Pi(W_8) = (w_5, w'_1)$. $\mathrm{K}_1\top$ and $\mathrm{K}_2\phi'_1$ are true in w_5 and w'_1, respectively, so $\mathrm{K}\phi'_1$ is true in W_8. Note that, although not shown in Table 4, $\mathrm{K}\top$ is true in all worlds.

Worlds in U_i are related to documents in D_i by way of the function $world_i$. Similarly, worlds in $U_\otimes$ are related to documents in $D_1 \cup D_2$ by way of a function $world_\otimes : (D_1 \cup D_2) \mapsto U_\otimes$. For $d \in D_1 \cup D_2$ and $W \in U_\otimes$, $W \in world_\otimes(d)$ iff $\mathrm{K}(property(d))$ is true W.

In our example, we obtain: $world_\otimes(d_1) = \{W_1, \ldots, W_5\}$, $world_\otimes(d_2) = \{W_1, \ldots, W_7\}$, $world_\otimes(d_3) = \{W_1, W_6, W_8\}$, $world_\otimes(d_4) = \{W_7\}$, and $world_\otimes(d_5) = \{W_2, W_3, W_6, W_7\}$.

Table 5. Most specific sentences and epistemic sets for the merged rank

ϕ	$e(\phi)$
$\phi_1 \wedge \phi'_1$	$\{W_1\}$
$\phi_1 \wedge \phi'_2$	$\{W_2, W_3\}$
ϕ_1	$\{W_4, W_5\}$
$\phi_2 \wedge \phi'_2$	$\{W_6\}$
$\phi_3 \wedge \phi'_2$	$\{W_7\}$
ϕ'_1	$\{W_8\}$
$\top$	$\{W_9, W_{10}, W_{11}\}$

Now that worlds composing $U_\otimes$ have been constructed, the most specific sentences for the combined ranking can be determined, as well as the corresponding epistemic sets $e_\otimes : S_\otimes \mapsto \wp(U_\otimes)$. Table 5 shows the sentence ϕ in $S_\otimes$ such that $e_\otimes(\phi) \neq \emptyset$. For example, for worlds W_2 and W_3, the most specific sentence is the conjunction of ϕ_1 and ϕ'_1. Thus $e_\otimes(\phi_1 \wedge \phi'_1) = \{W_2, W_3\}$.

A document in $d \in D_i$ has an associated set of most specific sentences $mss_i(d)$. Similarly a document $d \in D_1 \cup D_2$ has a set of most specific sentences. Let $mss_\otimes : D_1 \cup D_2 \mapsto \wp(S_\otimes)$ symbolising the set of most specific sentences for document in the combined ranking. For $d \in D_1 \cup D_2$, $mss_\otimes(d) = \{\phi \in S | e_\otimes(\phi) \subseteq world_\otimes(d)\}$.

4.3 Representing the Merged Ranking

As for single retrieval result, the combined ranking is defined in terms of a belief function defined upon a mass function. The mass function for the combined ranking is defined in terms of those of the individual rankings. We give first two definitions.

Definition 17 *Two sentences ψ and ϕ are* logically equivalent*, written $\psi \Leftrightarrow \phi$ iff $\psi \Rightarrow \phi$ and $\phi \Rightarrow \psi$.*

Definition 18 *The function $\Gamma : S_\otimes \mapsto \wp(S_1 \times S_2)$ maps every sentence ϕ in $S_\otimes$ to a subset of sentence pairs (ϕ_1, ϕ_2) with $\phi_i \in S_i$, $i = 1, 2$, such that $\phi_1 \wedge \phi_2 \Leftrightarrow \phi$ in $U_\otimes$.*

Γ relates sentences of $S_\otimes$ to those of S_1 and S_2.

In this work, we have assumed that the two retrieved sets are independent, which is the most common scenario in data fusion. With this assumption, the calculation of the mass function for the combined ranking is straightforward. However, it should be noted that the evidential reasoning as developed by Ruspini [11] allows for the dependent case to be taken into account.

Definition 19 *Let $m_\otimes : S_\otimes \mapsto [0,1]$ be the mass function associated with the combined ranking. Let $m_1 : S_1 \mapsto [0,1]$ and $m_2 : S_2 \mapsto [0,1]$ be the mass functions associated with agents* K_1 *and* K_2*, respectively. For $\phi \in S_\otimes$:*

$$m_\otimes(\phi) = \mathcal{K} \sum_{(\phi_1,\phi_2)\in\Gamma(\phi)} m_1(\phi_1) \times m_2(\phi_2)$$

where $\mathcal{K} = \sum_{(\phi_1,\phi_2)\in S_1\times S_2} m_1(\phi_1) \times m_2(\phi_2)$, ensuring that $m_\otimes$ is a mass function.

The next theorem ensures that $m_\otimes$ is a mass function which as for m_1 and m_2 assigns non-null value only to most specific sentences in $S_\otimes$.

Theorem 3 *Let $\phi \in S_\otimes$. We have $m_\otimes(\phi) > 0$ if $e_\otimes(\phi) \neq \emptyset$.*

To prove Theorem 3, we prove first the following two lemmas.

Lemma 2 *Let $\phi \in S_\otimes$ and $(\phi_1,\phi_2) \in \Gamma(\phi)$. If $e_\otimes(\phi) \neq \emptyset$, then $e_1(\phi_1) \neq \emptyset$ and $e_2(\phi_2) \neq \emptyset$.*

Proof: Let $\phi \in S_\otimes$ such that $e_\otimes(\phi) \neq \emptyset$. Let $(\phi_1,\phi_2) \in \Gamma(\phi)$: $\phi_1 \wedge \phi_2 \Leftrightarrow \phi$. Suppose that ϕ_1 and/or ϕ_2 are not most specific sentences in S_1 and S_2 ($e_1(\phi_1) = \emptyset$ and/or $e_2(\phi_2) = \emptyset$). Then for all worlds in U_1 and/or all worlds in U_2, there exit sentences $\psi_1 \in S_1$ and/or $\psi_2 \in S_2$ such that $\psi_1 \Rightarrow \phi_1$ and/or $\psi_2 \Rightarrow \phi_2$. For all worlds $W \in U_\otimes$ such that ϕ is true in W, from Definition 15, ϕ_1 and ϕ_2 are true in W and from Definition 16, ψ_1 and/or ψ_2 are true in W. Therefore, $\psi_1 \wedge \phi_2 \Rightarrow \phi_1 \wedge \phi_2$, $\phi_1 \wedge \psi_2 \Rightarrow \phi_1 \wedge \phi_2$, and/or $\psi_1 \wedge \psi_2 \Rightarrow \phi_1 \wedge \phi_2$. From the relation between ϕ_1, ϕ_2 and ϕ, any of the latter means that there always exist sentences ψ_1 and/or ψ_2 such that $\psi_1 \wedge \phi_2 \Rightarrow \phi$, $\phi_1 \wedge \psi_2 \Rightarrow \phi$, and/or $\psi_1 \wedge \psi_2 \Rightarrow \phi$. That is, ϕ is not a most specific sentence, which contradicts the hypothesis $e_\otimes(\phi) \neq \emptyset$. Therefore, $e_1(\phi_1) \neq \emptyset$ and $e_2(\phi_2) \neq \emptyset$. □

Lemma 3 *Let $\phi \in S_\otimes$. If $e_\otimes(\phi) \neq \emptyset$ then $\Gamma(\phi) \neq \emptyset$.*

Proof: Let $W \in U_\otimes$ and $\phi \in S_\otimes$. From Definition 15, for $\mathrm{K}\phi$ to be true in W, we have two sentences ϕ_1 and ϕ_2 in S_1 and S_2, respectively, such that $\phi_1 \wedge \phi_2 \Rightarrow \phi$. Furthermore, $\mathrm{K}\phi_1$ and $\mathrm{K}\phi_2$ are true in world W. However, ϕ is the most specific world in W, then it must be the case that $\phi \Rightarrow \phi_1 \wedge \phi_2$. Therefore, the set $\Gamma(\phi)$ cannot be empty if ϕ is a most specific sentence. □

We prove now Theorem 3.

Proof: Let $e_\otimes(\phi) \neq \emptyset$. From Lemma 3, $\Gamma(\phi) \neq \emptyset$. Let $(\phi_1,\phi_2) \in \Gamma(\phi)$. From Lemma 2, ϕ_1 and ϕ_2 are most specific sentences with respect to U_1 and U_2, respectively: $m_1(\phi_1) \neq 0$ and $m_2(\phi_2) \neq 0$. From Definition 19, $m_1(\phi_1) \times m_2(\phi_2)$ contributes toward the value of $m_\otimes(\phi)$ which is hence non-null: $m_\otimes(\phi) > 0$. □

The calculation of $m_\otimes$ for our working example is shown in Table 6. $\Gamma(\phi)$ contains only one pair (ϕ_1,ϕ_2). The normalising constant is $\mathcal{K} = 0.62$. The results in Table 6 show that, for instance, in the combined ranking, documents

Table 6. Mass function calculation

ϕ	ϕ_1	ϕ_2	$m_1(\phi_1) \times m_2(\phi_2)$	$m_\otimes(\phi)$
$\phi_1 \wedge \phi'_1$	ϕ_1	ϕ'_1	0.4×0.5	0.32
$\phi_1 \wedge \phi'_2$	ϕ_1	ϕ'_2	0.4×0.3	0.19
ϕ_1	ϕ_1	$\top$	0.4×0.2	0.13
$\phi_2 \wedge \phi'_2$	ϕ_2	ϕ'_2	0.3×0.3	0.15
$\phi_3 \wedge \phi'_2$	ϕ_3	ϕ'_2	0.2×0.3	0.096
ϕ'_1	$\top$	ϕ'_1	0.1×0.5	0.08
$\top$	$\top$	$\top$	0.1×0.2	0.032

Table 7. Belief function values for the merged set

d	$mss_\otimes(d)$	R(d)
d_1	$\{\phi_1 \wedge \phi'_1, \phi_1 \wedge \phi'_2, \phi_1\}$	0.64
d_2	$\{\phi_1 \wedge \phi'_1, \phi_1 \wedge \phi'_2, \phi_1, \phi_2 \wedge \phi'_2, \phi_3 \wedge \phi'_2\}$	0.89
d_3	$\{\phi_1 \wedge \phi'_1, \phi'_1, \phi_2 \wedge \phi'_2\}$	0.55
d_4	$\{\phi_3 \wedge \phi'_2)\}$	0.096
d_5	$\{\phi_1 \wedge \phi'_2, \phi_2 \wedge \phi'_2, \phi_3 \wedge \phi'_2\}$	0.44

characterised by $\phi_1 \wedge \phi'_1$ should be ranked before those characterised by $\phi_1 \wedge \phi'_2$. This result is plausible since from Table 3, we see that for ranking $\mathcal{R}_2$, ϕ'_1 characterises documents that should be ranked before those characterised by ϕ'_2.

The ranking of a document d is defined as for the single retrieval case in terms of a belief function defined upon $m_\otimes$ as $R(d) = Bel_\otimes(property_\otimes(d))$ where $Bel_\otimes$ is the belief function associated with $m_\otimes$. Applied to our working example, the belief values are shown in Table 7 (the sentences $mss_\otimes(d)$ are shown for each document d). The combined ranking is: d_2, d_1, d_3, d_5 then d_4. From the rankings $\mathcal{R}_1$ and $\mathcal{R}_2$, the possible combination of worlds from U_1 and U_2, and the mass functions m_1 and m_2, the combination of evidence as developed in evidential reasoning yields a coherent and intuitive combined ranking.

5 Conclusion

In this work, we have developed a *formal model* for data fusion based on evidential reasoning. The model considers the knowledge describing a retrieval result. The combination of knowledge determines the fusion of the retrieval results.

Our next step is the implementation of the model in order to study which properties of retrieved results lead to effective data fusion. The challenges will be (1) to map information and heuristics used so far in data fusion to properties, and (2) to determine appropriate mass functions that give higher values to properties that retrieve documents at higher rank. If the properties and the mass function are provided for each retrieval set, the combined ranking is a direct application of the proposed model.

References

1. Bartell, B., Cottrell, G., and Belew, R. Automatic combination of multiple ranked retrieval systems. In *Proc. 17th ACM SIGIR Conference* (Dublin,Ireland, 1994), pp. 172–181.
2. Baumgarten, C. A probabilistic model for distributed information retrieval. In *Proc. 21th ACM SIGIR Conference* (Philadelphia, USA, 1997), pp. 258–266.
3. Callan, J., Zhihong, L., and Croft, W. Searching distributed collection with inference networks. In *Proc. 18th ACM SIGIR Conference* (Seattle, USA, 1995), pp. 21–28.
4. Chellas, B. F. *Modal Logic: An introduction.* Cambridge University Press, 1980.
5. Dreilinger, D., and Howe, A. Experiences with selecting search engines using metasearchI. *ACM TOIS 15*, 3 (1997), 195–222.
6. Gauch, S., and Wang, H. Information fusion with ProFusion. In *Proceedings of the First World Conference of the Web Society* (San Francisco, CA, USA, 1996).
7. Gauch, S., Wang, H., and Gomez, M. ProFusion: Intelligent fusion from multiple distributed search engines. *Journal of Universal Computing 2*, 9 (1996).
8. Gravano, L., Chang, K., Garcia-Molina, H., and Paepcke, A. STARTS - Stanford protocol proposal for internet meta-searching. In *ACM SIGMOD* (1997).
9. Huibers, T. W. C. *An Axiomatic Theory for Information Retrieval.* PhD thesis, Utrecht University, The Netherlands, 1996.
10. Kantor, P., Ng, K., Hirsh, H., Basu, C., Loewenstern, D., and Kudenko, D. Data fusion of machine learning methods for the TREC-5 routing task (and other work). In *Proceedings of the fifth Text REtrieval Conference (TREC-5), NIST Special Publication* (1995).
11. Ruspini, E. H. The logical foundations of evidential reasoning. Tech. Rep. 408, SRI International, 1986.
12. Savoy, J., Calve, A. L., and Vrajitoru, D. Report on the TREC-5 experiment: Data fusion and collection fusion. In *Proceedings of the fifth Text REtrieval Conference (TREC-5), NIST Special Publication* (1995).
13. Selberg, E., and Etzioni, O. The MetaCrawler architecture for resource aggregation on the web. *IEEE Expert 12*, 1 (1997), 8–14.
14. Smeaton, A. Independence of contributing retrieval strategies in data fusion for effective information retrieval. In *Proceedings of the 20th BCS-IRSG Colloquium, Grenoble, France* (1998), Springer-Verlag Workshops in Computing. in press.
15. Smeaton, A., and Crimmins, F. Using a data fusion agent for searching the www. Poster presented at the WWW6 conference, 1997.
16. Tsikrika, T., and Lalmas, M. Merging techniques for performing data fusion on the web. In *Conference on Information and Knowledge Management (CIKM)* (2001), pp. 181–189.
17. Voorhees, E., Gupta, N., and Johnson-Laird, B. Learning collection fusion strategies. In *Proc. 18th ACM SIGIR Conference* (Seattle, USA, 1995), pp. 172–179.
18. Yager, R., and Rybalov, A. On the fusion of documents from multiple collection information retrieval systems. *JASIS 49*, 13 (1998), 1177–1184.

Flexible Management of Consistency and Availability of Networked Data Replications[⋆]

Francesc D. Muñoz-Escoí, Luis Irún-Briz, Pablo Galdámez,
José M. Bernabéu-Aubán, Jordi Bataller, M. Carmen Bañuls, and
Hendrik Decker

Instituto Tecnológico de Informática, Universidad Politécnica de Valencia
Camino de Vera s/n, E-46071 Valencia, Spain
{fmunyoz,lirun,pgaldam,josep,bataller,banuls,hendrik}@iti.upv.es

Abstract. We describe a family of three replication protocols, each of which can operate in three different modes of consistency. The protocols are tailored to satisfy the availability demands of interconnected databases that have a high degree of data locality. The protocols accomplish a grade of transaction completion which does not compromise availability, and ensure the consistency of replicas also if a transaction needs to be aborted. Flexibility of query answering is understood as optimizing the tradeoff between consistency and availability, i.e., between correctness and timeliness of query answering. This is achieved by choosing an appropriate protocol alternative, and changing the consistency mode of operation during the session, as appropriate for a given transaction.

1 Introduction

Consistency of replicated data is of the essence for query answering in distributed databases with multiple copies of data objects. Also a high availability of answers to queries is essential for many networked applications. However, the management of replication consistency may frequently need to lock data objects from access and therefore hamper their availability. Thus, the conflicting goals of consistency and availability have to be reconciled.

For tightly interconnected databases, consistency maintenance of replicated data is a fairly well-understood problem [1,2]. Traditional approaches for replicating databases usually deploy fast local area networks (LANs) [3,4,5,6,7], using network-intensive protocols. However, for less tightly connected (e.g., internet-based and, more generally, wide-area-networked) applications, the network tends to be a limited resource. This means that, for wide-area networks (WANs), the issues of maintaining consistency and availability need to be addressed differently from solutions for LANs [8,9,10].

Nowadays, many distributed (i.e., networked) applications have to manage very large amounts of data. Despite the increasing ubiquitousness of information, the access patterns to distributed data often feature a noticeable degree of

⋆ This work has been partially supported by the EU grant IST-1999-20997 and the Spanish grant TIC99-0280-C02-01

geographical locality. Moreover, many applications require a high grade of availability, in order to satisfy the need to offer services at any time, to clients that are either internal or external to the networked application. A predominance of locality of access patterns usually suggests a partitioning of the database [11, 12]. In many scenarios, it may be convenient or even necessary to replicate the information in a set of servers, each one attending its local clients. The different replicas must then be interconnected, a WAN being usually the best-fit architectural option.

The kind of applications we have in mind are, for instance, distributed databases in wide area intranets of medium and large enterprises, for data warehousing or resource management, as well as extranet service provisioning of such enterprises. Examples where such intranets of databases are deployed advantageously are enterprises with several branch offices (e.g., banks, chains of retailers, super- and hypermarkets), telecommunication providers, travel businesses, logistics, etc. Examples of extranet services which benefit from replication protocols as discussed in this paper are customer relationship management, e-banking, virtually all kinds of e-business as well as most e-government applications. Common to all of these applications is that potentially huge amounts of data are maintained and replicated on distributed sites, while access patterns (or at least significant contingents thereof) are highly local. Efficiency and high availability of such services is key to their acceptance and success.

In this paper, we propose three protocols to meet the consistency requirements of the applications just outlined. Although the protocols differ in the priorities of their particular goals, each of them is (more or less) optimistic (or, at least, cannot be classified as pessimistic), since transactions may proceed locally and are checked for consistency violation only at commit time. When a consistency violation is encountered, the transaction is rolled back. The tradeoff between consistency and availability, i.e., correctness and timeliness of query answering, is handled flexibly, by allowing the user to choose an appropriate protocol alternative for a given application, choose an appropriate consistency mode for each session, and change the consistency mode on-line to a more rigorous one, if the need arises, for a given transaction. In other words, flexibility of query answering is achieved by optimizing the tradeoff between the redundancy and the consistency of replicated data, so that answers are both available and sufficiently up-to-date at any time and anywhere in the network, despite occasionally broken links.

Section 2 delineates some characteristics that are common to each of the consistency protocols under consideration. Section 3 outlines an abstract consistency protocol and three particular implementations of this abstract model. In section 4, we give a detailed description of one of the consistency protocols outlined previously. Section 5 discusses how failures of transaction completion can be handled; in fact, failure handling is common to each of the three protocols. Section 6 describes the actions to be taken when a faulty node recovers. In section 7, we conclude with an outlook to further work and some final remarks.

2 Common Concepts

As a common feature of all protocols under consideration, we suppose that the database nodes communicate through local consistency agents which behave according to the protocols described later on. They can be viewed as mediators for every data access and each transaction performed by the local sessions.

Some of the protocols discussed below use lazy replication. Consequently, not all system nodes have the latest version of each data object. Thus, for each data object, each database node assumes one of the following roles:

– *Owner node*: Initially, that role is assumed by the node where the data object has been created. It handles *access confirmation requests* (see below) for the object; i.e., it allows or denies these requests when asked about it by the initiator of a session.
– *Synchronous nodes*: These are the nodes to maintain up-to-date replicas of the data object. For each object, the synchronous nodes and their number are supposed to be preconfigured. (A more advanced solution could implement a dynamic allocation of synchronous nodes for each individual object, but for simplicity, we assume a fixed allocation. Recall that replication redundancy is needed for supporting fault tolerance.)
– *Deferred nodes*: Such a node usually does not have an up-to-date replica of the object, although it may do so, sometimes (e.g., when it has initiated the session which has caused the latest change in the object).

Our approach to concurrency control takes into account that data may be replicated lazily, either due to prohibitive network performance or to avoid the freezing effects of using traditional locks. To each object, an owner node is assigned, for controlling accesses to that object. However, control is leveraged only when a session wants to commit.

When a session initiates its commit phase, the owners of the accessed objects need to check if the session has used the latest committed versions of the objects. In that case, the owners validate the accesses, and the session can effectively be committed. If any of the read objects is reported to have an outdated version, the session is flagged for abortion. We refer to these contact actions as "*access confirmation requests*", since they ask the owners to check the correctness of only those object versions that have been used in the given session.

In WANs, nodes may fail and network partitioning faults may happen. To cope with that, an appropriate notification service is needed. This service assigns static identifiers to each database node in the network. In general, consistency protocols require node identifiers for recording the object ownership, as well as the role assigned to each node when a particular data object is accessed.

Since we use *access confirmation request* management, each session identifier (SIDs) must include the identifier of the node that has initiated the session. Thus, in case of node failure, the consistency protocol knows which granted access requests did belong to the faulty node, thereby being able to handle the situation appropriately.

Similar to sessions, also the data objects themselves use an identification structure. For each object, an object identifier (OID) identifies its initial owner.

When a node fails, ownership of its objects is transferred to another node until it recovers again. However, this change of ownership has to be taken notice of by all alive nodes, such that the owner of each object can be identified correctly at any time.

Besides OIDs, also a version number needs to be associated to each object, for the purpose of node recovery. For recovering a database, the latter only needs to send to one of its neighbors the last known version number of each object it maintains. The receiving node then replies with all changes to be applied, to bring the recovering database back to an up-to-date state.

An object version number includes the SID of the session that has written its last value. That is needed to build the graph of causal precedent sessions in the protocol based on session updates (see sections 3.2 and 4 for more).

In case of a partitioning failure (i.e., when some network links break and only several isolated subgroups of nodes remain connected), access to an object is granted only if the subgroup containing that object contains a significant share (more than 50%) of alive nodes with up-to-date replicas of the object. (NB: A "*subgroup*" is a maximal subset of nodes remaining interconnected after a network partitioning failure, i.e., each subgroup is unable to communicate with any other.) If a partitioning failure occurs or if some node fails, the ownership roles of the faulty node are transplanted to one or several of the remaining alive nodes (if any) with replicas of the owned objects.

3 An Abstract Protocol Algorithm

All consistency protocols considered in this paper conform to a common basic algorithm consisting of the following steps:

1. For each database node, its consistency manager maintains a set of tables with meta-data. The meta-data comprise information about the versions of the locally stored objects.

   When a session tries to read-access an object, the local consistency manager checks its version in the database. If it detects the local version to be outdated, it sends a message to the owner node of that object, requesting the recent updates. The session is blocked until these updates have arrived and have been applied to the local database. Only then, the query that originated this read access can be executed.

   Write accesses are not managed this way. Rather, they are directly executed without any check. However, although no check is made, some meta-data records need to be removed. In general, for each access, the sets of objects read or written are entered into the meta-data tables in order to be later able to build the read- and write-set of the session.
2. When a session wants to commit, the local consistency manager retrieves the read- and write-sets of the session and sends an *access confirmation request* to each of the objects' owners, including the OID, the version number and the access mode (read or write) of each data object.

   When an owner node receives one of the above messages, it compares the object's version number with its latest version. If they are equal, the owner grants the requested access permission and sets the SID to be the tenant of the access grant. The grant is *exclusive* if the access mode was “write”, and it is *shared* if the access mode was “read”. The grant remains assigned to the requester until the session has propagated its updates to (at least) the synchronous nodes with replicas of the objects involved in the session. Once the session has been committed in this set of nodes, the session initiator changes the version number of the write-accessed objects in that session, and then releases the access grants.

If a session holds an access grant, other sessions requesting the same grant in a conflicting mode receive a service denial reply to their requests. Thereupon, the consistency manager which requested the conflicting grant aborts the corresponding session.

Using the approach sketched above, different consistency protocol variants can be designed for supporting different kinds of consistency modes. Next, we describe three such consistency modes. Later we show how different alternatives in the overall approach lead to those modes, and finally we outline the protocols resulting from such alternatives.

3.1 Three Different Consistency Modes

Different sessions may use different consistency modes. Modes may change after each session, and, for each transaction, a less restrictive mode may be promoted to a more restrictive one. The three modes are:

– *Plain*: This mode only allows isolated read requests and guarantees that all accessed objects respect a session-causal commit order. Thus, the version numbers may not be the latest ones; i.e., the accessed objects may be out of date.
– *Checkout*: This mode is a permissive variation of the transaction mode discussed below. It does not guarantee the isolation property. Thus, if several sessions perform read accesses to the same object, only one of them is allowed to promote its access to write mode. In some cases, this may break serializability, which in a standard transactional setting would lead to the blocking or the abortion of the promoting transaction. If two of these sessions promote their read accesses to write mode one of them is aborted.
– *Transaction*: The usual transaction properties of atomicity, consistency, isolation and durability must be provided.

Notice that we have described the consistency modes in increasing order of restrictiveness; i.e., transaction mode allows less concurrency conflicts than checkout mode, and checkout mode less than plain mode. So, when two sessions using different consistency modes are conflicting, some rules have to be adopted in order to resolve that conflict. In practice, priority is usually given to the more restrictive mode.

3.2 Consistency Protocols

We present three consistency protocols with different characteristics, as follows:

- *Full object broadcasting*: This protocol implements immediate updates of all database replications, so it does not use lazy replication. Thus, the write-set of a committed session is broadcast and applied to all database nodes immediately. Of course, not all sessions are committed, since object owners must grant the access confirmation to do so. These access permissions depend on the consistency mode used. The protocol is as flexible as to support each of the three consistency modes.
- *Simple object update*: This protocol uses lazy replication and object updates, instead of session updates. Although this protocol complies with all consistency modes, it requires more effort in plain mode, since the way updates are propagated in sessions using transaction or checkout mode does not entail the guarantees required for plain mode accesses.

  Note that, at commit time, the updates are only propagated to the preconfigured synchronous replicas of each modified object. Thus, the full effects of a session may possibly be not reflected at each node which has received an update message. That is, a node may have a synchronous replica of one of the objects involved, and a deferred replica of one of the other objects.

  When a read operation needs a more recent version than the one stored in the local database, only the latest version is requested (and obtained) from the object owner. No other contents need to be transferred.
- *Session set update*: This protocol uses lazy replication and session updates. That is, when the updates are transferred to other nodes, not only the object changes are transmitted to their synchronous replicas, but all session updates (i.e., the session write-set) are transferred to each node which has a synchronous replica of at least one of the changed objects.

  For supporting plain mode, an additional problem appears: before the effects of a session can be applied, all sessions preceding it in causal order need to have already been applied to the same database. Sometimes, however, this may have not yet been done. For instance, this may happen when an object has a deferred replica in a given node that has not received any update for a long period of time, and some of the objects with a synchronous replica in that node have been modified in the same session. Section 4 describes in more detail the Transaction Set Update consistency algorithm.

4 The Algorithm for the Session Set Update Protocol

This section specifies the algorithm of one of the proposed consistency protocols, viz. the session set update protocol, in more detail. For this protocol, we assume reliability of message transport, in the sense of TCP/IP reliability, as well as the existence of a notification service which informs alive nodes about failures and recoveries of others.

Each time a session commits, this protocol transfers the whole set of session updates. It is sent to each node which has a synchronous replica for at least one of the objects updated in that session. Consequently, plain mode can be easily supported, without further ado. That is, the plain-mode reads can be locally completed without any further message exchange. Each consistency manager executes the following algorithm:

- Every node maintains a log containing each session applied in its local database. A process for updating all nodes in the network is run asynchronously in each node. As soon as this asynchronous process has made sure that a session has been applied in each node, the session can be eliminated from the logs.
- When a node detects an out-of-date object N_o (say) in a read request, it locates the owner node of N_o and sends a request message to it in order to update its object copy. This request contains the identifier and version number of N_o.
- The owner node receives the request and looks at its meta-data for the set of causally dependent sessions, which are needed to update the requested object from the given version to the version held in the local database of the owner node. This process is performed by the following algorithm:
  - The owner node looks at the meta-data for the last session that modified the requested object T_o (say). In that session, other objects may have been read. For convenience, let us denote the read-set of T_o by $R(T_o)$.
  - For each object o_i in $R(T_o)$, the node should search its log for each session T_j which has o_i in its write-set (i.e., each T_j which causally precedes T_o).
  - The node then takes T_o and each preceding T_j for composing a graph representing the causal dependencies of T_o and each T_j.
  - Starting from T_o, the algorithm iterates inductively by layers of causal dependencies, in order to finally include all causal dependencies in the graph. The iteration ends when all the logged sessions with causal precedence have been included.
- The resulting graph (which actually contains statements of sessions) is sent to the requesting node. The latter analyzes the graph in order to eliminate already applied sessions, and to determine whether each session in the graph can be applied in its local database.

  A session S can be cut out of the graph when S has been already applied in the requesting node. This occurs when every object in the write-set of S has a lower version than the version held in the local database.

  It is important to note that a session T in the graph cannot be applied to the local database when an object contained in its read-set has a higher version number than the one held in the local database (i.e., there exists a causally precedent session which is yet unknown to the requesting node). For being in the position to apply T, the out-of-date object must first be updated.
- When the cutting out process is completed, the requesting node sends a new message to the owner node, requesting the complete write-set (values and version numbers) of the meta-session resulting from compacting the graph.

– The owner replies with a message containing the write-sets of each session included in the previous request. This information can be extracted from the meta-data tables, since all of these sessions have been locally applied by this owner node, which therefore knows about all of these write-sets.

Note that "plain" consistency mode is easily implemented. It ensures that each update preserves causal consistency. The following is needed to provide this functionality:

– A log of every applied session for each node. Note that this may imply a redundancy of logs.
– A session is kept logged until an asynchronous process has checked that the session has been applied to each node.

5 Failure Handling

Several failure scenarios must be considered for ensuring that these protocols are fault-tolerant. Subsequently, we are going to discuss two kinds of such situations. The first adds more detail to the steps given in section 4. The second deals with the migration of ownership roles between nodes.

5.1 Failure Handling in the Protocol's Algorithm

When a node fails, the assumed notification service monitors this failure and informs all alive nodes about it. Partition failures are notified the same way, but then, the set of faulty nodes may be larger.

Let us see what happens with the sessions initiated by a faulty node. We distinguish the following cases, according to the step at which the failure occurs:

– If one of these sessions has not yet surpassed the access confirmation granting step of the algorithm, no record of that session exists in any of the alive nodes. So, that session can be discarded. And when its host node recovers, it must abort that session, forcing its application to repeat the work.
– If the session fails once it has obtained the remote access grants, but before it has multicast any update, a similar situation arises. No record of the session updates can be found in any of the live nodes, so the session cannot be terminated in the remaining nodes. Consequently, that session must be aborted when its host node recovers. However, the faulty node has obtained some access grants, which may prevent other sessions from going ahead.

  To avoid this, we propose the following. If an object update has been received by one of its synchronous replicas, all of them have received this update, because the multicasts are atomic and reliable. So, when the notification service communicates a node's failure, all object owners scan their grants lists. If access to some object has been granted to a session initiated in the faulty node, the access granter (i.e., the owner of the object) has to check if some update multicast associated to the SID which requested that access has been received. If no such update was received, the grant can be released. Otherwise, the following point has to be considered.

– If the session has at least initiated the update multicast, its updates may have arrived to other nodes. In that case, we need to perform the same actions as in the previous case. The grants held by this session have to be released. Since the updates have been received (which may occur only if all grants have been obtained and all changes have been made in the original node but are still not committed), no additional access grant is needed. Since the session initiator node has failed, the grants are not needed by any other node which would replace the faulty one, because such a replacer node has already committed this session. Consequently, the access grants have already been used correctly and thus must be released now. With that, the session is completed.

  No other case needs consideration. Possibly, the session had not been completed yet, but this only means that it held some grants that have been released as a consequence of the steps explained above.

– Another problem arises in the following situation. Suppose the node has failed once the update multicast for an object was initiated, but before all deferred replicas holding the latest version number so far have been notified that this version has changed and that they therefore do not have an up-to-date version in their local database any more. Then, the problem is that the node which eventually "inherits" the object ownership does not have the correct information about the up-to-dateness of the deferred replicas.

  When the node assigned to be the object owner assumes its new role, some actions must be performed. It already knows that it maintains a synchronous replica. It may decide that one or several other nodes should be updated to hold a synchronous replica. (This decision depends on the particular replication policy to be implemented in a given scenario.) Moreover, it has to broadcast a message to each accessible node which holds a deferred replica of an object whose ownership it has "inherited". This message contains the version number of the newly owned objects. The deferred nodes will reply to this message by indicating whether their current replica is actually out-of-date or not.

5.2 Role Migration

When a node fails, each object it owns has to be managed by one of the remaining alive nodes. Thus, object ownership has to be migrated to one of those other nodes. So, we have to discuss two tasks in this section. The first one deals with the criterion for electing the node replacing the faulty one. The second task deals with the migration of the access granting management.

– Election of the replacer node is based on the static identifiers associated to the preconfigured members of the system. Since each node has such an identifier, it suffices to choose the alive node holding a synchronous replica with the lowest identifier among all identifiers that are greater than that of the faulty node (or the lowest one, if the faulty node had the greatest identifier). To be able to elect the new owner, a significant share of synchronous replicas must still exist in the system.

  In case of using an even number of synchronous replicas, a particular criterion needs to be determined for breaking the tie in case of network partitioning. For instance, the subgroup that holds the node with lowest identifier among the previous set of synchronous replicas may be determined to contain the new owner.

  We assume that the number of synchronous replicas is known in advance. In case of network partitions, it may happen that the owner for a given object remains alive, but the greater part of its synchronous replicas remain unavailable. In that case, the current owner must give up its role. Then, the following problem may arise. If the majority of synchronous replicas are in another subgroup after the network partition, one of them will take up the management role, according to the strategy proposed in the previous paragraph. However, if those replicas have failed, no other session in the whole system will be able to use that object again.

– The node which "inherits" the object ownership has to ask each other node about the access grants they have; i.e., it has to know which of the grants it manages has an owner and who is that owner. To this end, each node holding a grant that was managed by the faulty node, sends an ownership message to the new owner. This message contains the identifier of the object associated to the grant, the access mode, and the SID of the session that holds it. If a node has no grants, it sends an empty message.

  A timeout for receiving all of these messages is set by the new owner. If some message has not been received timely, an explicit request is sent to that node.

  Note that the criterion used to select the replacer node is known to all consistency managers, so no message is needed to identify the new owner. This is also true in case of network partitions. If one subgroup loses an owner for a given object, no session accessing that object will be allowed in that subgroup. Moreover, all grants maintained in that subgroup have to be released, and all sessions having had those access grants must be aborted.

6 Node Recovery

When a node recovers, the notification service informs all alive nodes about it, and the system state is reconfigured. That, is, the recovering node resumes ownership for each object it had initially created. Recall that one of the alive nodes had intermediately "inherited" the object ownership for all objects of the recovering node. Upon recovery of the original owner, the intermediate owner must send all information concerning access grants to the recovering node. As long as this message is not received by the recovered node, no incoming access grant requests or releases can be processed. Rather, they have to be buffered, and once the message and the object ownership have been transferred to the recovered node, the latter resumes the access grant management.

It may happen that, for some object, an access request or release still reaches its intermediate owner node, right after the latter has passed back the ownership to the original node. In that case, each such message must be forwarded to the original owner. The sender of the message does not need to take care of this; forwarding is done by the intermediate owner, who knows the current (original) owner node.

Special attention is needed in case the recovered process belongs to the class of nodes that must maintain synchronous replicas of several objects. The owners of such objects have to re-include it in the set of synchronous replicas, and possibly one of those replicas has to be degraded to the deferred category (although it initially maintains an up-to-date copy, but eventually it will become obsolete if it is degraded). No extra message exchange is needed to do so.

Note that, during reconfiguration, no new sessions are allowed to be initiated, and the session management is temporarily disabled. Further note that the recovered node needs to receive all relevant updates having taken place during its down-time. It requests these updates from one of its alive neighbors (e.g., the one to which object ownership had most likely been transferred to, previously). This request includes the OIDs and version numbers of objects in its local database. As a reply, its neighbor sends a message with all updates needed.

7 Conclusion

Query answering in distributed systems is increasingly important to a large number of networked applications. The availability and up-to-dateness of answers is endangered by network failures, but can be prevented by replicating data and taking measures to maintain their consistency. However, maintaining replication consistency often needs to lock data, thus obstructing again their availability. In this paper, we have proposed a flexible reconciliation of the conflicting goals of availability and consistency, i.e., of timeliness and correctness of query answering, by providing a choice among different but compatible replication protocols, which can be chosen appropriately for each application, for each session and for each transaction.

Concurrency control in each of the discussed protocols is fairly optimistic, in the sense that access confirmation requests may be made only when local updates have terminated. This behavior is appropriate for applications which predominantly work with "local" data, i.e., data that has been created on site (possibly by other applications pertaining to the same location), but where replication is needed to improve the access to "remote" data, most of which do not require more than "read only" access.

The algorithms discussed in this paper are currently being implemented within the GlobData project [13]. GlobData is a software tool which provides an object-oriented view of replicated relational databases. GlobData's architectural environment is conceived to support different kinds of applications with various access patterns. By offering different replication modes with several protocol options (including other protocols that have not been described here [14]), GlobData also supports the tuning and evaluation of performance optimizations for each particular application.

The kind of appliances for GlobData we have focused on so far are in the realm of applications as characterized in the introduction. In future, we intend to widen our scope by also investigating distributed database networks for mobile applications. For example, the scheduling of agendas for salespersons and managers on the move is an application with a lot of business potential. Asynchronous consistency maintenance protocols for replicated data objects that are distributed over nomadic networks have been proposed, e.g., in [15], and it should be interesting to assimilate them into the GlobData framework, as well as assess them in a way we have done for the protocol options discussed in this paper.

References

1. Bernstein, P., Hadzilacos, V., Goodman, N.: Concurrency control and recovery in database systems. Addison-Wesley (1987)
2. Wolfson, O., Jajodia, S.: Distributed algorithms for dynamic replication of data. In: Proc. of the 11th Symposium on Principles of Database Systems, San Diego, CA, USA (1992) 149–163
3. Kemme, B., Alonso, G.: A suite of database replication protocols based on group communication primitives. In: International Conference on Distributed Computing Systems. (1998) 156–163
4. Krishnakumar, N., Bernstein, A.J.: Bounded ignorance: A technique for increasing concurrency in a replicated system. ACM Trans. on Database Sys. **19** (1994) 586–625
5. Jajodia, S., Mutchler, D.: Dynamic voting algorithms for maintaining the consistency of a replicated database. ACM Trans. on Database Sys. **15** (1990) 230–280
6. Herlihy, M.: Apologizing versus asking permission: Optimistic concurrency control for abstract data types. ACM Trans. on Database Sys. **15** (1990) 96–124
7. Herlihy, M.: Dynamic quorum adjustment for partitioned data. ACM Transactions on Database Systems **12** (1987) 170–194
8. Ladin, R., Liskov, B., Shrira, L., Ghemawat, S.: Providing high availability using lazy replication. ACM Trans. on Comp. Sys. **10** (1992) 360–391
9. Ferrandina, F., Meyer, T., Zicari, R.: Implementing Lazy Database Updates for an Object Database System. In: Proc. of the 20th International Conference on Very Large Databases, Santiago, Chile (1994) 261–272
10. Ferrandina, F., Meyer, T., Zicari, R.: Correctness of lazy database updates for object database systems. In: POS. (1994) 284–301
11. Rahm, E.: Empirical performance evaluation of concurrency and coherency control protocols for database sharing systems. ACM Trans. on Database Sys. **18** (1993) 333–377
12. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proc. of the International Conference on Management of Data, Montreal, Canada (1996) 173–182
13. Instituto Tecnológico de Informática: Globdata web site. Accessible in URL: `http://globdata.iti.es` (2002)
14. Rodrigues, L., Miranda, H., Almeida, R., Martins, J., Vicente, P.: Strong replication in the globdata middleware. In: Proc. Workshop on Dependable Middleware-Based Systems, pp. G96-G104, Washington D.C., USA (Suplemental Volume of the 2002 Dependable Systems and Networks Conference, DSN 2002). (2002)
15. Alonso, L.: Optimistic data object replication for mobile computing. In: 9th IFIP/IEEE Workshop on Distributed Systems: Operations and Management. (1998)

Querying Multidatabase Systems Using SIQL

N. Parimala and T.V. Vijay Kumar

School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi 100067 India
{parimala_n, tvvijaykumar}@hotmail.com

Abstract. Several approaches have been proposed to provide access to data which is spread across different sources. Prominent among these are the approaches taken in federated systems and multidatabase systems. The main thrust is to provide the user with a mechanism to access several databases in a unified way. However, in all these systems the user is not isolated from specifying the path expressions in the query. In our system, we propose a query language, Structure Independent Query Language (SIQL), which has no reference to the structure of the databases. The structure is arrived at by the system by using the component databases that are registered with it. In general, there can be multiple queries that can be generated for a user query. We define 'closeness' criterion which imposes a ordering on the queries that can be generated. The queries are generated and executed according to this ordering and the result is presented to the user.

1 Introduction

Several systems have been proposed to provide integrated access to users where the data is spread over many sources. Two major approaches have been taken to define the distributed system – the global schema approach and the multidatabase language approach. In the global schema approach the heterogeneous schema descriptions available at different sites are combined and a single unified global schema is arrived at, as in SDD-1[27], MULTIBASE[18], POREL[22], SIRIUS DELTA[19], UniSQL/M[14]. The associated mapping information between the global schema and the set of databases is defined. The queries are specified by the user using this unified structure. This approach is adopted in federated databases and global-schema multidatabases [25], [24], [9], [7], [26], [3], [6].

The mediator-wrapper approach [32] is similar to global schema approach in that it provides the user with a virtual database by reconciling the differences between its wrapped data sources. Queries are posed in a declarative language to this virtual database. TSIMMIS[11] has adopted this approach.

On the other hand, in multidatabase language systems there is no global schema. The systems provide query language tools to integrate information from different databases. That is, the integration responsibility now rests with the user and the user must be knowledgeable about the local data representations. This knowledge is used in framing the queries, as in MDSL[20], MSQL[21], HOSQL[1], OSQL[8], SchemaSQL[17].

Immaterial of whether a global schema is defined or not the user has to be aware of the structure of the database(s). The query languages of the above systems expect the user to use this knowledge of the structure of the database in framing the query. The user has to explicitly state the relation from which the data is to be retrieved. In case of a conflict in the relation name, the relation name has to be qualified with the database name. It must be noted that the above problem of having to specify the query using database structure exists even in single database systems. However, this gets compounded when we are dealing with more than one system. There is a need to lessen the user burden while specifying the query.

In this paper we take a different approach to querying databases. We believe that the user need not be aware of the structure of the schema in order to query the database. We provide the user with a query language which has no overt reference to the structure of the database. That is, the query is specified independent of all path expressions. It is up to the system to determine the structures from which the data is to be fetched. These structures are inferred at run-time using the repository of the databases registered with the system. Using these structures the path expressions are inserted by the system in the user query. The generated query is subsequently executed.

In general more than one query with path expressions can be generated for a given user query. As an example consider the schemas given in Figure 1 of two component DBSs which may be at two different sites.

Schema_1	**Schema_2**

Employee(E#,Ename,Job,D#)
Department(D#,Dname,Company,Cloc)
	Company(C#, Cname ,Cloc)
	Department(D#, C#, Dname)

Fig. 1.

Let the user query be

```
SELECT Ename, Dname
```

The above query can be answered using the following two resolutions

a) the Employee and Department relations of Schema_1, and
b) the Employee relation of Schema_1 and Department relation of Schema_2.

The corresponding queries[10] are

```
a) SELECT Schema_1.Employee.Ename, Schema_1.Department.Dname
    FROM Schema_1.Employee, Schema_1.Department
   WHERE Schema_1.Employee.D# = Schema_1.Department.D#
```

```
b) SELECT Schema_1.Employee.Ename, Schema_2.Department.Dname
   FROM Schema_1.Employee, Schema_2.Department
   WHERE Schema_1.Employee.D# = Schema_2.Department.D#
```

In our proposal the result of executing each generated query is presented to the user one after the other. The question that naturally arises is as to which of the generated queries is the most desired one from the user's point of view. If we look at the example above, we find that the user might prefer resolution (a) to (b). This is because resolution (a) and the corresponding generated query involves lesser number of schemas. This preference is expressed using the notion of 'closeness'. If the generated query involves only one relation, then it is very 'close'. At the other extreme is the query which spans maximum schemas and maximum relations. That is, the generated queries which involve lesser number of relations (and therefore, lesser joins) and lesser schemas are more 'close' than those which involve more joins and schemas. This 'closeness' criterion can be used to order the generated queries with the ones more close higher in the order.

We propose to generate queries in the order defined by 'closeness'. Once a query is generated, it is executed and the result is presented to the user. If the user so desires the generated query is displayed so that the user knows the schemas and relations that have been used to answer the user query. The user is given the option to move to the next query and so on till all the queries are exhausted.

In our system the schemas which participate may or may not correspond to the same reality. Therefore, it may happen that the attributes with the same name but belonging to different schema relations have different meanings or have the same meanings. This aspect is not addressed as we do not integrate the schemas. Each result provided to the user describes one meaning for the attribute.

The layout of this paper is as follows. Section 2 describes the user interface. Conflict resolution is dealt with in the next section. The notion of redundant queries is explained in section 4. Section 5 defines the 'closeness' criterion and the ordering of the queries. The last section is the concluding section.

We use the schemas given in Figure 2 for the rest of the paper.

Schema_1
 	
Employee(E#, Ename, Job, D#)
Department(D#, Dname, Company, Cloc)
	Schema_2
	
	Company(C#, Cname, Cloc)
	Department(D#, C#, Dname)

Schema_3
 	
Industry(Company, Department, Cloc)
Branch(Br#, Cloc, Cname, Dname)
	Schema_4
	
	Employee(E#, Department, C#)
	Company(C#, Cname, Cloc)

Fig. 2.

2 The User Interface

A GUI interface is built which helps the user to frame the query and is also responsible for presenting the result to the user.

2.1 The User Query

In our system, the user is not expected to know the structure of the databases. Therefore, a GUI interface is built which presents the user with all the names that are defined in the various databases that are registered with the system. This interface helps the user in forming the query. The details are given in [31].

The query formed by the user is akin to SQL. Many query languages have been defined in the literature. Instead of adding one more, we have adapted SQL[13] which is an industry standard.

As specified earlier, the query must contain no explicit reference to the structure of the database. The query is expressed using the SELECT clause but with the FROM clause deleted. In other words, the form of the query in SIQL is as follows:

SELECT name$_1$, name$_2$ … name$_n$
WHERE condition

The condition is a boolean expression which is specified using AND, OR and NOT. Further, no attribute is prefixed with the relation name or the database name followed by the relation name. Similarly the names in the condition are qualification free. Notice that our query language has no path specification.

2.2 Presenting the Result

As seen above, the generated queries involve different path expressions for the same name in different schemas. These different expressions will yield different results. For example, in the query

SELECT Department

the name Department has many mappings. It can be resolved to the relation name in Schema_1 and Schema_2; as an attribute of relation Industry in Schema_3 and as an attribute of relation Employee in Schema_4. The four queries correspondingly will be

1) SELECT D#, Dname, Company, Cloc FROM Schema_1.Department
2) SELECT D#, C#, Dname FROM Schema_2.Department
3) SELECT Department FROM Schema_3.Industry
4) SELECT Department FROM Schema_4.Employee

The result in the first case is a relation with order 4, in the second case a relation of order 3, a relation with a single attribute in the third and the fourth instances.

There are two ways in which the result can be presented to the user. In the first case, many results are presented to the user. These are as many as the number of different resolutions. In the above example, there are four results and of order 4, 3, 1 and 1 respectively.

The second method would be to present to the user just one set of tuples. The strategy adopted is as follows. For each set containing the result the missing attributes are added to the set in such a way that the sets become union compatible. The result for these added attributes is set to null. The union of all the sets is performed. That is, the outer union of all the sets is taken. For example, the answer to the above query would have six attributes

D#, Dname, Company, Cloc, C#, Department

The attributes C# and Department would have null values whereever there is value for the first four attributes and so on. This option, we feel, may not be the right way of presenting information to the user. For example, in the case when there is no common attribute among the sets, there would be nulls appearing in many fields of many tuples. This in effect is presenting the result of the first query followed by the second and so on. The resulting relation may not be a true reflection of the data. We have therefore, chosen to present the different resolutions separately. That is, we have adopted the first option. In the above example there would be four results. Each result is presented to the user one after the other by the GUI interface.

3 Conflicts

Semantic heterogeneity occurs when the same data is expressed in more than one schema differently[12]. There can be conflict in meaning or intended use between attributes which have the same syntactic structure. These arise when data defined in different databases are to be combined to give a unified view. In some cases it may be possible to define simple unions or outer unions whereas in some others there may be no universal key. These issues have been dealt with in [2], [15], [16], [5], [28], [29], [23].

In our approach, for every resolution only one intended use of an attribute is picked up. As a result, the different meanings do not run into a conflict in the same resolution. The different meanings of a given attribute will appear in different resolutions and hence, in different answers.

4 Redundant Queries

Some query resolutions are redundant. In other words, they generate the same data as other queries. Consider, the following query

```
SELECT   Company, Department
WHERE   Cloc = 'Bangalore' OR Dname = 'Computer'
```

Consider two generated queries.

```
SELECT  Schema_2.Company.*, Schema_2.Department.*
FROM    Schema_2.Department, Schema_2.Company
WHERE   Schema_2.Company.C# = Schema_2.Department.C# AND
        ( Schema_2.Company.Cloc = 'Bangalore' OR
            Schema_2.Department.Dname = 'Computer' )
```

and

```
SELECT  Schema_2.Company.*, Schema_2.Department.*
FROM    Schema_2.Department, Schema_2.Company, Schema_3.Industry
WHERE   Schema_2.Company.C# = Schema_2.Department.C# AND
        Schema_2.Company.Cloc = Schema_3.Industry.Cloc AND
          ( Schema_2.Company.Cloc = 'Bangalore' OR
            Schema_2.Department.Dname = 'Computer' )
```

The data selected in the second query is identical to the data retrieved in the first query. This is because after taking the equi-join of Company and Industry, there is no additional data retrieved from Industry. The second query here is a redundant query. In our system the redundant queries are not generated.

5 The Two Principles

When the user specifies a query, as shown above, there can be more than one resolution for a given user query. We believe, that the users prefer certain resolutions over others. If all the names are resolved to a single relation of a schema, then that resolution is the most preferred one. If that is not possible, then the next 'close' resolution is the one where all of them belong to one schema. Within these, the query resolution with lesser number of joins is more 'close' than the one with more joins. If none of these is possible, then data from multiple schema is to be retrieved. This desirability is expressed using two principles - density and relationship. These principles are explained below.

The 'density' principle states that if a field has multiple definitions then, the resolution of a name to the structure that has been referred to maximum number of times in this query resolution is more close than others which have been referred to lesser number of times. As an example, consider the query

```
SELECT Company, Cname, Dname, Cloc
```

Let us assume that Company has been resolved to Schema_3.Industry and Cname and Dname have been resolved to Schema_3.Branch. Cloc can be resolved to either Schema_3.Industry or Schema_3.Branch. Schema_3.Industry has been referred to once in this query resolution whereas Schema_3.Branch has been referred to twice. Therefore, resolution of Cloc to the latter is more close than resolution to the former.

The 'relationship' principle states that if a field has multiple definitions then the resolution of a name to a structure in the query which gives rise to minimum number of joins is more close than that structure where the number of joins are more. As an example, consider the query

```
SELECT E#, Cname
```

Let us assume that E# has been resolved to in Schema_4.Employee. The resolution of Cname to Schema_4.Company is a one-way join whereas the resolution of Cname to Schema_3.Branch is a two-way join - first between Schema_4.Employee and Schema_3.Industry and then between Schema_3.Industry and Schema_3.Branch. The former is more close than the latter.

In order to incorporate 'closeness', we define two terms - Distance and Weighted Count. Weighted count, WC, is a value associated with every query and it incorporates the 'density' principle. Distance incorporates 'relationship' principle. These terms are explained below.

In a generated query, all the names in the user query are prefixed with the schema name and the relation name. With each relation R that is present in the generated query, we associate a 'reference count'. The reference count gives the number of times R has been referred to in the generated query.

WC is computed as follows:

Let the user query be of the form

SELECT $name_1$, $name_2$ ……… $name_i$
WHERE $name_{i+1}$ = ' ' AND $name_n$ = ' '

Here, the number of names in the user query is n. Let the relations which are referenced in the query resolution be R_1, R_2 … R_m. Let their reference counts be C_1,C_2 … C_m ordered in the descending order of reference counts. Then,

$$\text{Weighted Count WC} = \sum_{i=1}^{m} (n - i + 1)Ci$$

WC gives a measure of the density.

The distance for a generated query is computed as follows:

Distance = number of joins * number of schemas

Distance incorporates 'relationship' principle.

5.1 Query Resolution Order

Applying the principles of density and relationship, the queries can be arranged in order of 'closeness' with the ones more close higher in the order. The resolutions which have minimum number of relationships (joins) are higher in the order of 'closeness'. Within resolutions which have the same number of joins, the ones in which density is maximized are more close than others.

In terms of WC and Distance, this can be translated as follows:

1. First order the generated queries according to Distance starting with minimum Distance.
2. Within queries having equal Distance, order the queries in the descending order of Weighted Counts.

In the above ordering, the query with minimal Distance and maximum WC is the most 'close' query.

5.2 An Example

Consider an example. Let the query be

```
SELECT   Company, Department
WHERE   Cloc = 'Bangalore' OR Dname = 'Computer'
```

The complete resolution set for the above query contains 74 queries. We demonstrate the manner in which distance and WC is computed for a couple of these. Let a generated query be

```
SELECT   Schema_2.Company.*, Schema_2.Department.*
FROM     Schema_2.Company, Schema_2.Department
WHERE    Schema_2.Company.C# = Schema_2.Department.C#  AND
         ( Schema_2.Company.Cloc = 'Bangalore' OR
           Schema_2.Department.Dname = 'Computer' )
```

The names in the user query is 4. So n = 4. The relations that are referenced are Schema_2.Department and Schema_2.Company. Therefore, m = 2. The reference count for Schema_2.Department is 2 - it is referenced in the select clause and Dname is an attribute of Schema_2.Department. The reference count of Schema_2.Company is again 2 as it is referenced in the select clause and Cloc is an attribute of Schema_2.Company. In the formula

$$\mathrm{WC} = \sum_{i=1}^{m} (n - i + 1)Ci$$

m is 2, C1 is 2 and C2 is 2. Therefore,

$$\mathrm{WC} = \sum_{i=1}^{2} (4 - i + 1)Ci = 14$$

In order to compute the distance we have to consider the number of joins and the number of schemas. The number of schemas is 1 and the number of joins is also 1. Therefore,

Distance = number of joins * number of schemas = 1

Consider, now, another query resolution for the same user query above.

```
SELECT   Schema_3.Industry.Company, Schema_3.Industry.Department
FROM     Schema_2.Company, Schema_2.Department, Schema_3.Industry
WHERE    Schema_3.Industry.Cloc = Schema_2.Company.Cloc AND
         Schema_2.Company.C# = Schema_2.Department.C# AND
         ( Schema_2.Company.Cloc = 'Bangalore' OR
           Schema_2.Department.Dname = 'Computer' )
```

The relations that are referenced are Schema_3.Industry, Schema_2.Department and Schema_2.Company. The reference count for Schema_3.Industry is 3 as it is appearing twice in the select clause and once in the WHERE clause. The reference count for Schema_2.Department is 2 as Dname and C# are attributes of this relation. The Reference count of Schema_2.Company is 2 as Cloc and C# in the WHERE clause are attributes of Company. In the formula

$$\mathrm{WC} = \sum_{i=1}^{m} (n - i + 1)Ci$$

n is 4, m is 3, C1 is 3, C2 is 2 and C3 is 2. Therefore,

$$\mathrm{WC} = \sum_{i=1}^{3} (4 - i + 1)Ci = 22$$

In order to compute the Distance we have to consider the number of joins and the number of schemas. The number of schemas is 2, Schema_2 and Schema_3 and the number of joins is also 2. Therefore,

Distance = number of joins * number of schemas = 4

As mentioned earlier, there could be a number of redundant queries. For the example query above, 67 queries are redundant. The remaining 7 queries with the WC and Distance is shown in Figure 3.

5.3 Implementation

In the preceding sections we ordered the query resolutions. However, if all the resolutions are first worked out and the queries are later ordered, the algorithm will be polynomial in time. In our system the queries are generated in the desired order. The actual details of the algorithm are given in [31].

We have used JAVA as the programming language for writing Client programs and Server programs and RMI for communication between distributed JAVA applications. The system uses ORACLE[10] as back-end and connection to the databases is established using JDBC.

6 Conclusion

In this paper we have described a system for accessing data spread across several sources. The emphasis has been to provide the user with a query language wherein the query can be expressed without any explicit reference to the schema.

The different databases register with the system. The names of the relations and the attributes form the namespace for the queries. The query has a SELECT and a WHERE clause. As the names in the namespace are unknown to the user, we have developed a GUI interface which helps the user frame the query.

The user query is translated to a SQL query by resolving the names and the actual structures to which they belong. It has been shown that there are many resolutions for a given query. Principles of density and relationship have been defined which impose an ordering on the queries. The queries are generated according to this order and are presented to the user.

We have measured the performance of the system by measuring the increase in the time taken to generate and execute the generated query. Graphs were plotted by varying the number of schemas and names in the user query. The graphs show that as we increase the number of schemas from four to eight there is no significant increase in time[31]. This shows that the system performance does not degenerate as the number of schemas increases.

Generated Query	**Distance**	**Weighted Count**
SELECT Schema_1.Department.*, Schema_1.Department.Company FROM Schema_1.Department WHERE (Schema_1.Department.Cloc='Bangalore' OR Schema_1.Department.Dname='Computer')	0	16
SELECT Schema_3.Industry.Company, Schema_3.Industry.Department FROM Schema_3.INDUSTRY, Schema_3.Branch WHERE Schema_3.Industry.Cloc=Schema_3.Branch.Cloc AND (Schema_3.Industry.Cloc='Bangalore' OR Schema_3.Branch.Dname='Computer')	1	18
SELECT Schema_2.Company.*, Schema_2.Department.* FROM Schema_2.Company, Schema_2.Department WHERE Schema_2.Company.C#=Schema_2.Department.C# AND (Schema_2.Company.Cloc='Bangalore' OR Schema_2.Department.Dname='Computer')	1	14
SELECT Schema_4.Company.*, Schema_2.Department.* FROM Schema_4.Company, Schema_2.Department WHERE Schema_4.Company.C#=Schema_2.Deparment.C# AND (Schema_4.Company.Cloc='Bangalore' OR Schema_2.Department.Dname='Computer')	2	14
SELECT Schema_4.Company.*, Schema_4.Employee.Department FROM Schema_3.Branch, Schema_4.Company, Schema_4.Employee WHERE Schema_3.Branch.Cloc=Schema_4.Company.Cloc AND Schema_4.Company.C#=Schema_4.Employee.C# AND (Schema_3.Branch.Cloc='Bangalore' OR Schema_3.Branch.Dname='Computer')	4	16
SELECT Schema_4.Employee.Department, Schema_3.Industry.Company FROM Schema_2.Department, Schema_3.Industry, Schema_4.Employee WHERE Schema_2.Department.C#=Schema_4.Employee.C# AND Schema_3.Industry.Department=Schema_4.Employee.Department AND (Schema_3.Industry.Cloc='Bangalore' OR Schema_2.Department.Dname='Computer')	6	20
SELECT Schema_2.Company.*, Schema_4.Employee.Department FROM Schema_2.Company, Schema_3.Branch, Schema_4.Employee WHERE Schema_2.Company.Cloc=Schema_3.Branch.Cloc AND Schema_2.Company.Cname=Schema_3.Branch.Cname AND Schema_2.Company.C#=Schema_4.Employee.C# AND (Schema_3.Branch.Cloc='Bangalore' OR Schema_3.Branch.Dname='Computer')	6	16

Fig. 3.

We propose to enhance our system by providing a method by which different query results can be combined. This then, would dynamically define a global integrated virtual view and a virtual query which is posed for this view.

It may be argued that if the user has to decide which query he/she really wants then the user is forced to know the schemas and their structure. It must be noted, however, that it is far easier if somebody were to insert the path expressions in a query than if it is to be done by the user explicitly. In addition, if new schemas register with the system then new queries are automatically generated by the system involving these new schemas.

Our system is different from others where attempts have been made to ease the burden of the user. For example, in FINDIT[4] the user is assisted to formulate a query which is spread across several databases and coalitions. This is a form of query refinement where the user who is unaware of the distribution is helped to locate the database of interest. Similarly, in [30] the GUI interface assists the user by finding the necessary joins and generates the corresponding query upon confirmation from the user. On the other hand, in our system we present the user with all the possibilities.

Acknowledgement. The second author wishes to acknowledge the CSIR, New Delhi, India, for their financial assistance.

References

1. Ahmed, R., Smedt, P., Du, W., Kent, W., Ketabchi, A., Litwin, W.: The Pegasus Heterogeneous Multidatabase System. IEEE Computer, December (1991)
2. Batini, C., Lenzerini, M., Navathe, S.B.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys, Vol. 18, No. 4, December (1986) 323-364
3. Bergamaschi, S., Castano, S., Vincini, M.: Semantic Integration of Semistructured and Structured Data Sources., SIGMOD Record, Vol.28, N0. 1, March (1999) 54-59
4. Bouguettaya, A., King, R., Zhao, K.: FINDIT : A Server Based Approach to Finding Information in Large Scale Heterogeneous Databases., In Proceedings of 1st International Workshop of Interoperability in Multidatabases Systems, Kyoto, Japan, (1991) 191-194
5. Bright, M.W., Hurson, A.R., Pakzad, S.: Automated Resolution of Semantic Heterogeneity in Multidatabases. ACM Transactions on Database Systems, Vol. 19, No. 2, June (1994) 212-253
6. Castano, S., De Antonellis, V., Di Vimercati, S. De. C.: Global Viewing of Heterogeneous Data Sources., IEEE Transactions on Knowledge and Data Engineering, Vol. 13, No. 2, March/April (2001) 277-297
7. Chang, C.C..K., Garcia-Molina, H., Paepcke, A.: Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System., ACM Transactions on Information Systems, Vol. 17, No. 1, January (1999) 1-39
8. Chomicki, J., Litwin, W.: Declarative definition of Object-Oriented Multidatabase Mappings. In M.T. Ozsu, U. Dayal and P. Valduriez editors, Distributed Object Management. M. Kaufmann Publishers, Los Altos, California, (1993)
9. Chung, S.M., Mah, P.S.: Schema Integration for Multidatabases using Unified Relational and Object-oriented model., CSC '95, Proceedings of the 1995 ACM 23rd Annual Conference on Computer Science, February 28 - March 2, 1995, Nashville, TN, USA. ACM, (1995) 208-215

10. Durbin, J., Bobrowski, S., Vasterd, P.: Oracle8™, Distributed Database Systems, Release 8.0, Part No. A58247-01, Copyright © 1996, 1997, Oracle Corporation, December (1997)
11. Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The TSIMMIS Approach to Mediation : Data Models and languages, Journal of Intelligent Information Systems, 8(2), (1997) 117-132
12. Hull, R.: Managing Semantic Heterogeneity in Databases : A Theoretical Perspective. In Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, (1997) 51-61
13. International Organization for Standardization (ISO) : Information Technology - Database Languages-SQL-Technical Corrigendum 2, Document ISO/IEC 9075: 1992/Cor.2:1996(E) (1996)
14. Kelley, W., Gala, S.K., Kim, W., Reyes, T.C., Graham, B.: Schema Architecture of the UniSQL/M multidatabase system, In Modern Database Systems, (1995)
15. Kim, W., Seo, J.: Classifying Schematic and Data heterogeneity in Multidatabases Systems., IEEE Computer, 24(12), (1991) 12-18
16. Kim, W., Choi, I., Gala, S., Scheevel, M.: On resolving Semantic Heterogeneity in Multidatabase Systems., Distributed and Parallel Databases, 1(3), (1993) 251-279
17. Lakshmanan, L.V.S., Sadri, F., Subramanian, I.N.: SchemaSQL - A Language for Interoperability in Relational Multidatabase Systems. Proceedings of the 22nd VLDB Conference, Mumbai(Bombay), India, (1996)
18. Landers, T., Rosenberg, R.L.: An overview of MULTIBASE. in Distributed Databases, H.-J. Schneider, Ed., North Holland, The Netherlands, (1982) 153-184
19. Litwin, W., Boudenant, J., Esculier, C., Ferrier, A., Glorieux, A., Chimia, J.L., Kabbaj, K., Moulinoux, C., Rolin, P., Stangret, C.: SIRIUS Systems for Distributed Data Management. In Distributed Data Bases, H.J. Schneider, Ed. North-Holland, The Netherlands, (1982) 311-366
20. Litwin, W., Abdellatif, A.: An Overview of the Multi-Database Manipulation Language MDSL. Proceedings of the IEEE, Vol. 75, No. 5, May (1987) 621-632
21. Litwin, W., Abdellatif, A., Zeroual, A., Nocolas, B., Vigier, P.: MSQL : A Multidatabase Language. Information Sciences, 49, (1989) 59-101
22. Neuhold, E.J., Walter, B.: An Overview of the Architecture of the DDBs POREL. in Proc. Symp. Distributed Databases, Berlin, 1982, H.J. Schneider Ed., Amsterdam, The Netherlands : North Holland, (1982) 247
23. Parent, C., Spaccapietra, S.: Issues and Approaches of Database Integration. Communications of the ACM, Vol. 41, No. 5, May (1998)
24. Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J., Widom, J.: Querying Semistructure Heterogeneous Information. In International Conference on Deductive and Object Oriented Databases, (1995) 319-344
25. Reddy, M.P., Prasad, B.E., Reddy, P.G., Gupta, A.: A Methodology for Integration of Heterogeneous Databases., IEEE Trans. on Knowledge and Data Engineering, Vol. 6, No. 6, December (1994) 920-933
26. Rezende, F. de F., Hermsen, U., de sa Oliviera, G., Pereira, R.C.G., Rutschlin, J.: A Practical Approach to Access Heterogeneous and Distributed Databases. CAISE'99, LNCS 1626, (1999) 317-332
27. Rothnie, J.B., Bernstein, P.A., Fox, S., Goodman, N., Hammer, M., Landers, T.A., Reeve, C., Shipman , D.W., Wong, E.: Introduction to a System for Distributed Databases (SDD-1). ACM Transactions on Database Systems, Vol. 5, No. 1, (1980) 1-17
28. Sheth, A.P.: Semantic Issues in Multidatabase Systems. SIGMOD Record, Vol. 20, No. 4, pp. 5-9, December (1991)
29. Sheth, A., Kashyap, V.: So Far(Schematically) yet So Near(Semantically)., Proceedings IFIP WG 2.6 Conference on Semantic of Interoperable Database Systems(Data Semantics 5), (1993) 283-312

30. Tan, W.C., Wang, K., Wong, L.: QUICK : Graphical User Interface to Multiple Databases., Proceedings of the 7th International Workshop on DB & Expert System Application, (1996)
31. Vijay Kumar, T.V., Parimala, N.: Generation of Queries for Multidatabase Systems using SIQL (communicated).
32. Wiederhold, G.: Mediators in the architecture of future information systems., IEEE Computer, 25(3), (1992) 38-49

Extending Relational Data Access Programming Libraries for Fuzziness: The `fJDBC` Framework

Miguel-Angel Sicilia[1], Elena García[2], Paloma Díaz[1], and Ignacio Aedo[1]

[1] DEI Laboratory, Computer Science Department, Carlos III University
Avda. de la Universidad, 30, 28911 Leganés, Madrid, Spain
{msicilia,pdp}@inf.uc3m.es, aedo@ia.uc3m.es
http://www.dei.inf.uc3m.es
[2] Computer Science Department, Alcalá University
Ctra. Barcelona km. 33.600, 28871 Alcalá de Henares, Madrid, Spain
elena.garciab@uah.es

Abstract. Fuzzy relational databases have been extensively studied in recent years, resulting in several models and representation techniques, some of which have been implemented as software layers on top of diverse existing database systems. Fuzzy extensions to query languages and end-user query interfaces have also been developed, but the design of programming interfaces has not been properly addressed yet. In this paper, we describe a software framework called `fJDBC` that extends the *Java Database Connectivity API* by enabling fuzzy queries on existing relational databases, using externally-stored metadata. Since the main design objective of this extension is *usability* for existing database programmers, only a restricted subset of extensions (supported also by an extended object modelling notation) has been included. The overall design of the framework and some empirical results are also described.

1 Introduction: Programming Interfaces for Extended-for-Fuzziness Databases

1.1 Motivation and Related Work

The research area of fuzziness in database management systems (DBMS) has resulted in a number of models aimed at the representation of imperfect information in databases, or at enabling non-precise queries (often called *flexible queries*) on conventional database schemas. The former include both object-oriented models [6] and relational models (in which the two major approaches are similarity/proximity relation-based models [5,15], and possibility distribution-based ones [4,13,18], along with generalized models that combine them like `GEFRED` [11]). In addition, flexible querying is in many cases achieved by extending the syntax and semantics of the `SQL` language, as, for example, in `SummarySQL` [14].

These models extend existing systems and have been applied in real-world situations. However, as a matter of fact, no widespread commercial implementation of a *'native'* fuzzy database management system (FDBMS) currently exists.

Since internally extending commercial relational systems seems to be a difficult to justify effort for database vendors (despite of the fact that even fuzzy physical database access mechanisms have been proposed, for example [20]), the approach taken in many cases is that of building some kind of software layer on top of an existing DBMS (for example, Sonayst's `Fuzzy Query` [1] product can add flexible querying capabilities to any `ODBC`-compliant DBMS).

In this work, we focus on extending programming interfaces, rather than on extending the declarative `SQL` language or end-user querying capabilities. Our ultimate aim is that of devising an easy-to-understand and useful set of fuzzy database constructs delivered as an extension of existing database programming libraries. Our target programming interfaces are `JDBC` libraries, although our approach can be easily migrated to similar database access frameworks like Microsoft's `ActiveX Data Objects` (`ADO`). We have called our libraries `fJDBC` ('f' standing for fuzzy), since they're closely tied to `JDBC` libraries and adhere strictly to their programming semantics.

Other related work in the specific field of relational systems includes extensions to small desktop database engines, like `FQUERY` for Microsoft's `Access`[2] [21], and specific extensions for large relational DBMS, like `FSQL` for `Oracle`[3] [8].

1.2 Design Principles

Sun's `JDBC` is an application programmer's interface (API) that provides access to relational databases through existing or specific drivers in Java programs [4]. Although the latest `JDBC` specification (version 3.0) has adopted some of the features found in the `SQL99` standard, our framework only supports `SQL92` ones.

Since the ultimate aim of our research is to effectively bring fuzzy programming to common database software developers, we have departed from the human understanding of the existing `JDBC` interfaces, and not from existing fuzzy database models. This approach tries to accommodate only those fuzzy modelling notions that are close to the semantics of the classes that are part of the API, and it is not concerned with achieving maximum coverage of the diverse fuzzy database constructs. Our work represents a novel approach in fuzzy extensions of relational databases due to the following facts:

– We focus on application programmer's technology, and not on command-based or graphical query interfaces, so that the task model of the database programmer is the driving design input.
– As a result of its design, it can be used with any `JDBC`-compliant database (this includes the vast majority of commercial systems) by adding some simple *external* metadata, allowing for a fuzzy (re-)interpretation of existing legacy databases.

[1] http://fuzzy.sonalysts.com/
[2] http://www.microsoft.com/office/access
[3] http://www.oracle.com
[4] http://java.sun.com/products/jdbc/

– It was also designed to give explicit support to conceptual notions that arise in some extensions of fuzzy conceptual modelling, specifically, it allows the implementation of some forms of extended-with-fuzziness `Unified Modeling Language` (`UML`) static diagrams sketched in [16].

The main design objective of our work is *usability*[5], and more specifically, the following general principles or heuristics:

– *Consistency*, that is, avoiding the break of the original `JDBC` semantics.
– *Simplicity*, minimizing the number of added programming constructs.
– *Ease of learning*, measured in terms of time to master the new features of the interfaces.

The design of `fJDBC` follows a principle of "minimal change" from existing database programming structures, namely, minimal added interfaces and/or operations and minimal extensions to query languages. In this sense, we have designed the extension as a thin software layer on top of `JDBC`.

The rest of this paper is organized as follows. Section 2 describes the entities that can be represented in `fJDBC` and how they are implemented. Section 3 describes the API and sketches how enhanced `JDBC` queries can be issued against these entities. Section 4 briefly describes some relevant facts about the implementation, and conclusions and future work are provided in Section 5.

2 `fJDBC` Underlying Model

We'll describe the basic underlying `JDBC` model using Chen's notation [7], where a *n-ary* relation scheme on domains $D_1 \dots D_n$ is denoted as $R(A_1 \dots A_n)$ where each A_i is an attribute with domain $dom(A_i) = D_i$. A tuple t in relation R has therefore the form $t = (t(A_1) \dots t(A_n)) \in D_1 \times \dots \times D_n$ with $t(A_i)$ being a value $t(A_i) \in D_i$. One or several of the attributes in R form its primary key, denoted as $PK(R)$ (we'll use only one attribute called *oid* as primary key to simplify notation from here on). In addition, we'll denote by $FK(R)$ the attributes that form the *foreign key* in a relation R' with respect to table R. In what follows, common object model to relational schema mappings [3] are used.

2.1 The Fuzzy-Relation Model in `fJDBC`

The Fuzzy Relation Model [1] relaxes the relational framework by allowing a tuple to belong to a particular relation in a degree between [0,1], instead of in the classical {0,1} set, so that a membership function is (implicitly) defined on each relation R, in the form $\mu_R : D_1 \times \dots \times D_n \to [0 \dots 1]$. This model (often referred to as 'fuzziness at the object or tuple level') extends the notion of *'result set'* that is present in all current database access libraries. `UML` models can be used to represent this kind of vagueness by using class' stereotypes as showed in Figure 1.

[5] Ease of use was also one of the objectives of `JDBC` design

The case in Figure 1 correspond to a class representing a single fuzzy set or category. Three alternative implementation approaches can be used in `fJDBC` (examples for the first two are depicted in Figure 1):

1. Using an *special attribute* storing the belonging grade of every tuple in its relation, resulting in an extended relation $R'(A_1 \ldots A_n, m)$, with $m \in [0..1]$ representing the belonging grade.
2. Storing the fuzzy subset in a separate relation $R'(FK(R), m)$ leaving the original relation R unchanged, and modelling its relationship with its crisp domain as a standard referential integrity constraint.
3. Using a class that acts as a reified function $f : R \rightarrow [0..1]$ that yields a real value in the unit interval from an expression on the database schema. This classes must implement the `FuzzySet` interface which provides for this task an operation `getMembership(Object oid)`.

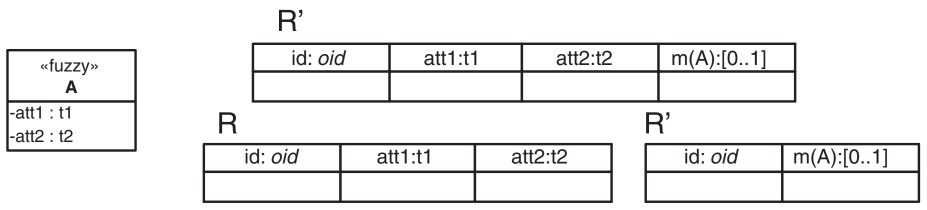


Fig. 1. `UML` Models for fuzziness at the object level

The second approach has the benefit of not requiring modification to existing table schemas and supports the two fuzzy relation scenarios in Figure 1, but it has performance drawbacks in some situations, as will be described later. The latter approach is the least intrusive in terms of schema update, and could be implemented through *stored procedures* in relational systems that provide this facility, although a more database-independent approach would be desirable. Currently, first, second and a restricted version of the third approaches are implemented in `fJDBC`, as will be described later.

The case in Figure 2 represent fuzzy subsets of an existing set (that in turn may be crisp or fuzzy) through a specialization relationship. Generalization semantics are extended from that of the `UML` metamodel, so that if an instance belongs with a non-zero grade to a class, it also belongs with a non-zero grade to all of its ancestors. In addition, the degree that an instance belongs to a fuzzy subclass C should not be greater than the degree that the instance belongs to any of its superclasses (as proposed in [7]). That is, given that C is a fuzzy subclass of B, $\mu_C(e) \leq \mu_B(e) \;\; \forall e \in C$.

Fuzzy specializations are implemented by storing the fuzzy subset in a separate relation, modelling its relationship with its domain as an (enhanced) relational integrity constraint. In the case of implementing a generalization between

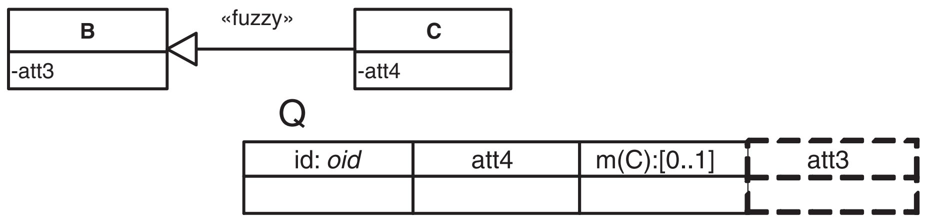


Fig. 2. Fuzzy Generalization as a derived Fuzzy Relation

superclass B and subclass C, we use an additional relation that includes the primary key of the parent entity: $Q \in D_{PK(B)} \times [0..1] \times D_{C1} \times \ldots \times D_{Cm}$, with D_{Ci} being the *i-th* attribute of C (in the case of multiple inheritance, additional foreign keys and membership attributes would be added). This way, when we refer to entity C in queries, all the attributes inherited from ancestors are implied, e.g. a tuple corresponding to entity C in Figure 2 will have the virtual form: $t = (t(PK(B)) \times t([0..1]) \times t(att3) \times t(att4))$, with $t([0..1])$ being the membership value of the tuple in C. It should be noted that, in terms of efficiency, this approach is similar to the second one for fuzzy relations. Subclasses can also be defined by including an additional membership field in the relation representing the parent class, but this approach can only be used in single-inheritance cases.

2.2 Associations and the Similarity-Based Framework in fJDBC

A *crisp* relation represents the presence or absence of interconnectedness between the elements of two or more sets. This concept is referred to as *association* when applied to object oriented modelling. According to the UML ([12], Semantics section, 2-20), an association defines a semantic relationship between classifiers, and the instances of an association can be considered a set of tuples relating instances of these classifiers, where each tuple value may appear at most once. A binary association may involve one or two fuzzy relations (unidirectional or bidirectional), although due to the semantic interpretation of associations, they're considered to convey the same information (i.e. the association between authors and books is interpreted in the same way despite the navigation direction). Fuzzy relations are generalizations of the concept of crisp *relation* (not to be confused with the term relation in the database sense used in the previous section) in which various degrees of strength of relation are allowed [10]. A binary fuzzy relation R on $X \times Y$ is a fuzzy subset of that cartesian product:

$$R = \{((x, y), \mu_R(x, y)) | \, (x, y) \in X \times Y\}$$

The concept of fuzzy association can be specified at the conceptual level through an stereotype on UML associations, as showed in the left part of Figure 3 – in this case, applied to an association class instead of a simple association.

In addition, an specialization `<<similar>>` of the `<<fuzzy>>` stereotype can be used to express similarity relations between instances (the special properties of this relations are handled by the software layer). Note that this kind of similarity approach differs from previous ones [7] in that similarity is specified only at the tuple level (and not at the attribute level).

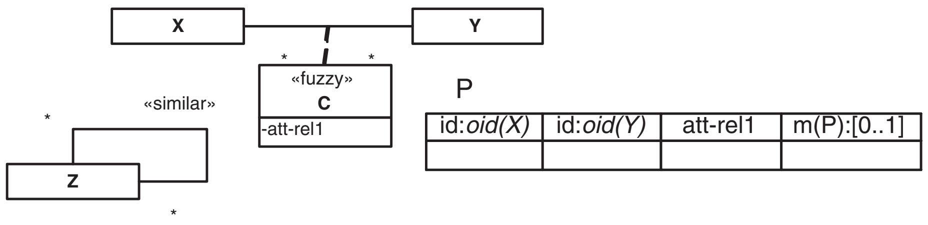


Fig. 3. Fuzzy Associations in `UML` Models

`fJDBC` supports binary fuzzy associations between fuzzy or crisp entities. The representation in the relational schema uses an intermediate relation $P \in D_{PK(X)} \times D_{PK(Y)} \times [0..1] \times D_{P1} \ldots D_{Pm}$ where D_{Pi} is the *i-th* association attribute (if any).

3 Framework Description

In this section, we describe the general aspects of the data access mechanism (subsection 3.1), the meta-data definition mechanism (subsection 3.2.) and how queries that involve imprecision are interpreted and results are obtained (subsection 3.3).

3.1 Extending Existing Drivers through Delegation

`JDBC`-driver implementers must provide a small footprint class that implements the `java.sql.Driver` interface. A driver class is responsible for its self-registration by providing an instance of itself to `java.sql.DriverManager` class. Our solution is that of providing a wrapper instance (of class `FuzzyDriver`) for an existing `JDBC` driver and delegating behavior to the internal instance, as an application of the *Proxy* design pattern [9]. Our driver can be loaded into a *Java Virtual Machine* (JVM) just as any other one, for example, by explicitly loading the class:

```
try{
    Class.forName("es.uc3m.inf.dei.fjdbc.FuzzyDriver");
}catch(ClassNotFoundException e)
    {System.out.println(e);}
```

The above typical fragment of code loads the driver class, causing the registration of a driver instance (through a Java `static` code fragment inside

FuzzyDriver). It should be noted that the JDBC driver that will be wrapped by our driver is internally loaded in the same fashion as a result of connection establishment.

Connections are then obtained from the DriverManager class by using a special JDBC protocol, as showed in the following code fragment.

```
Connection con=null;
try{
  con = DriverManager.getConnection("jdbc:fuzzy:prueba.xml",
                                   "", "");
  }catch(Exception e)
          {e.printStackTrace(); ...}
```

The JDBC protocol we have defined begins with the jdbc:fuzzy specifier followed by an string that should indicates the location of the metadata file for the database. That location must be specified as a currently a relative path pointing to the location of an XML file [19]. This approach will be subject to change in the near future to a more flexible one that uses an URI [2] instead, but for now, using a XML file provides a simple, readable and easily modifiable way to specify schema configurations.

The connection returned by the DriverManager in response to fJDBC connection strings are instances of the FuzzyConnection class, which implements the java.sql.Connection interface. In turn, result sets are obtained from a class FuzzyStatement that implements the java.sql.Statement interface. Results are thus represented by instances of the FuzzyResultSet class, which implements the java.sql.ResultSet interface, and adds methods to it to retrieve the compatibility grade of the tuples returned by queries involving fuzziness. A summary of the essential connection and query management classes that adhere to the set of interfaces defined in JDBC for the purpose of extension is showed in Figure 4, where UML dependencies stereotyped with <<returns>> indicate that one or several of the methods return an instance conforming to the target interface/class. Note that metadata are stored at connection level, as a set of FuzzyRelationMetadatum items, which are also accessible in FuzzyStatement instances created by the connection.

3.2 Metadata Specifications

To illustrate the capabilities of the model, we'll follow the example that is depicted in Figure 5, which can be considered a *user model* of a Web *e-commerce* application that involve fuzziness [17]. We have a set of users defined as fuzzy – due to some imperfect process, like user session reconstruction from Web usage mining, that is irrelevant for our discussion –, with three (fuzzy) subsets:

- The subset of sporadic users, characterized by a low level of sessions per time period (represented by the derived attribute numSessionsMonth).
- The subset of premium users, characterized by a rather high level of sessions per time period, and that have placed at least an order. These users have an attribute that stores the initial date in which they become *premium*.

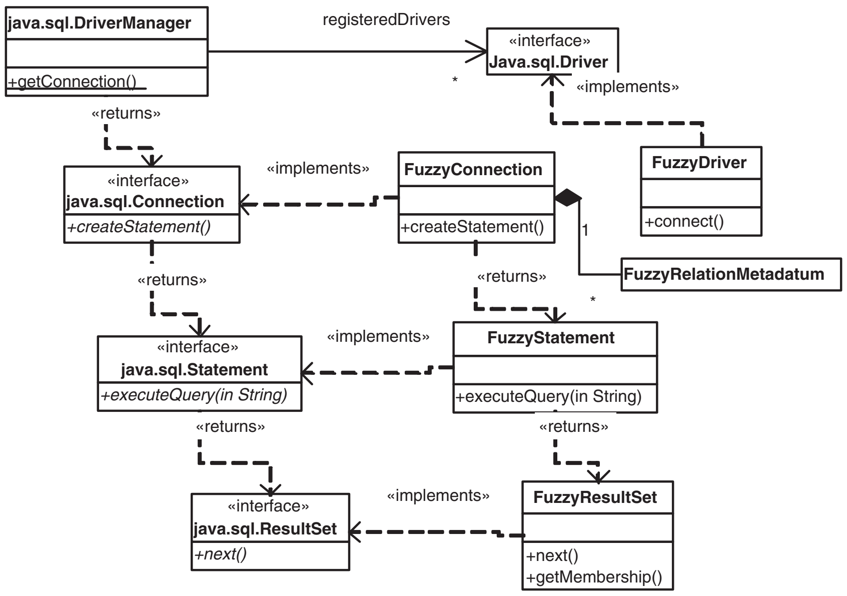


Fig. 4. Connection and query result management classes in `fJDBC` and their relationship with `JDBC` ones

– The subset of `loyal` users, that, in addition to being `premium` users, place a number of orders regularly (represented by the `numOrdersMonth` derived attribute). These users can accumulate `points` for commercial promotions.

In addition, the systems registers the (graded) interest in predefined subjects of each user (`interestedIn` association), and those subjects have a certain similarity with others (`related` similarity association).

The previously mentioned metadata `XML` file contains information about the storage of fuzzy relations. The example in Figure 5 could be defined in a file with the following contents:

```
<?xml version="1.0" encoding="UTF-8"?>
<fuzzy:db
xmlns:fuzzy="http://www.dei.inf.uc3m.es/fjdbc">
    <fuzzy:driver name="sun.jdbc.odbc.JdbcOdbcDriver"/>
    <fuzzy:connString name="jdbc:odbc:users"/>
    <fuzzy:fuzzy-relation name="users">
        <fuzzy:relation  name="users"/>
        <fuzzy:oidField  name="id"/>
        <fuzzy:membField name="users-m"/>
    </fuzzy:fuzzy-relation>
    <fuzzy:fuzzy-relation name="sporadic" type="subclass">
        <fuzzy:relation  name="users"/>
        <fuzzy:oidField  name="id"/>
```

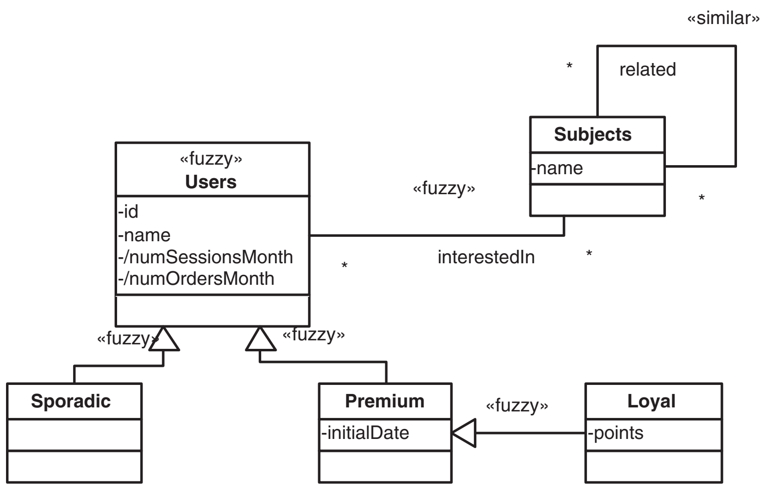


Fig. 5. Example conceptual model for a database of users of a Web site

```
    <fuzzy:membField name="sporadic-m"/>
</fuzzy:fuzzy-relation>
<fuzzy:fuzzy-relation name="premium" type="subclass">
    <fuzzy:relation name="users" oidField="id"/>
    <fuzzy:intermediate name="premium"
                    foreignOID="oid-user"
                    membField="premium-m"/>
</fuzzy:fuzzy-relation>
<fuzzy:fuzzy-relation name="loyal" type="subclass">
    <fuzzy:relation name="premium"/>
    <fuzzy:domain class= "es.uc3m.inf.dei.test.Loyal"/>
    <fuzzy:intermediate name="loyal" foreignOID="oid-premium">
</fuzzy:fuzzy-relation>
<fuzzy:fuzzy-association name="interestedIn">
    <fuzzy:endOne name="users" oidField="id"/>
    <fuzzy:endTwo name="subjects" oidField="id"/>
    <fuzzy:intermediate name="interested"
                     foreignOID1="oid-user"
                     foreignOID2="oid-subject"
                     membField ="interest-m"/>
</fuzzy:fuzzy-relation>
<fuzzy:fuzzy-association name="related" type="similarity">
    <fuzzy:endOne name="subjects" oidField="id"/>
    <fuzzy:endTwo name="subjects" oidField="id"/>
    <fuzzy:intermediate name="related"
                     foreignOID1="oid-subject1"
                     foreignOID2="oid-subject2"
```

```
                            membField ="related-to"/>
    </fuzzy:fuzzy-relation>
</fuzzy-db>
```

A fuzzy database (`fuzzy-db`) is described by specifying the `JDBC` connection string `connString` and the `driver` of the underlying relational database (the schema of that database is supposed to be created yet), along with a set of metadata items described by a list of (fuzzy) relation and/or association tags. For brevity's sake, we'll not describe the entire schema in detail, but sketch how definitions are specified:

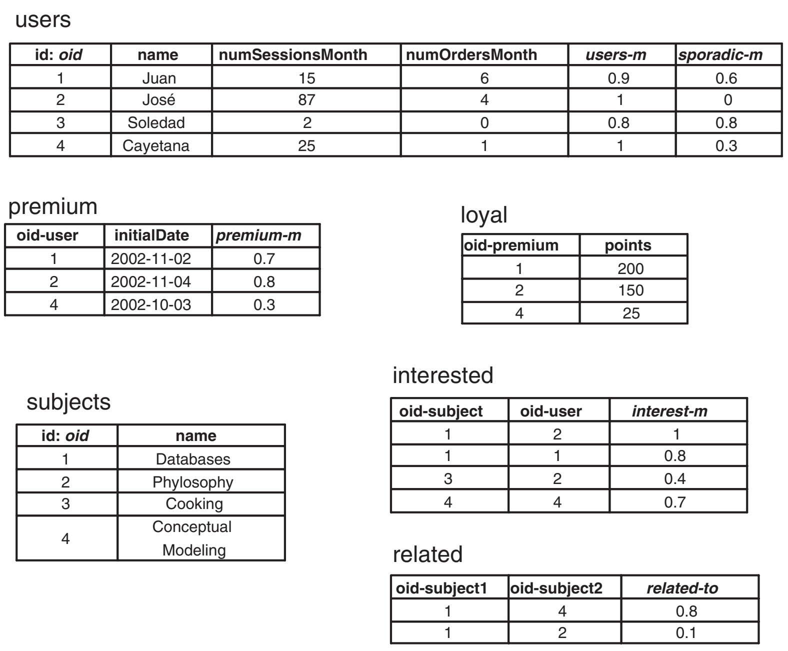
– The `users` relation is defined as a fuzzy relation on a table of the same name with a membership field `users-m`.
– The `sporadic` relation is defined as a subclass of the `users` relation, in the same `users` relation, through a `sporadic-m` field. The memberships are lower or equal than those of the superclass, and they would be defined by some fuzzy set on `numSessionsMonth` that is not considered here.
– The `premium` relation is defined by using an additional table, containing its `initialDate` attribute and its corresponding membership field.
– The `loyal` relation is defined in an additional table, but the membership values of its elements are obtained from an external class specified in the `domain` tag. The `Loyal` class will yield a membership value depending on the attribute values of its parent classes, and their own attribute values. Thus, the function is *hard-coded* in the class.
– The association between users and subjects is also defined, with no relation-specific attributes by specifying the intermediate table that holds the membership of the corresponding pairs of foreign keys.
– The similarity association is defined as an association (reflexive in this case). Note that only a number of the similarity values are actually stored. Default values for the rest of the subject pairs are computed by the software layer, based on the reflexivity, symmetry and *max-min* transitivity properties that are commonly defined on fuzzy similarity relations.

Figure 6 shows the example relational schema along with some illustrative sample data (note that this scheme would have additional tables not related to fuzziness, like, for example, those to store the entire, detailed purchase history).

3.3 Queries and Iteration

Once the schema is defined, queries can be issued through the same mechanisms that are allowed in `JDBC`. As a design decision, the programmer will see fuzzy relations and associations as additional tables (i.e. views), so that he/she issues `SQL`-syntax common queries, and the membership grades are retrieved transparently. As a result, the `SQL` syntax is not extended, but re-interpreted. The following extended queries are currently implemented:

– On a single fuzzy relation or a set of related fuzzy and crisp relations in the `FROM` clause. In this case, a *T-norm* is used to yield the membership value of



users

id: *oid*	name	numSessionsMonth	numOrdersMonth	*users-m*	*sporadic-m*
1	Juan	15	6	0.9	0.6
2	José	87	4	1	0
3	Soledad	2	0	0.8	0.8
4	Cayetana	25	1	1	0.3

premium

oid-user	initialDate	*premium-m*
1	2002-11-02	0.7
2	2002-11-04	0.8
4	2002-10-03	0.3

loyal

oid-premium	points
1	200
2	150
4	25

subjects

id: *oid*	name
1	Databases
2	Phylosophy
3	Cooking
4	Conceptual Modeling

interested

oid-subject	oid-user	*interest-m*
1	2	1
1	1	0.8
3	2	0.4
4	4	0.7

related

oid-subject1	oid-subject2	*related-to*
1	4	0.8
1	2	0.1

Fig. 6. Example relational scheme and data for the user database

the tuples in all the relations, as in `SELECT name FROM premium, sporadic`. In this case, the result set will include all the users that belong (i.e. that has nonzero membership in both subgroups. If we use the minimum as *T-norm*, we'll obtain the result set `rs =` {`'Juan'/0.6, 'Cayetana'/0.3`}

– On a fuzzy subclass, which retrieves all the attributes in its ancestors (which can be used in the formulation of the query). For example, in `SELECT * FROM loyal WHERE initialDate>='2002-6-12'`, attribute `initialDate` is defined in an ancestor.
– On fuzzy associations, by simply giving the name of the association. For example, the query `SELECT users.name, subjects.name FROM interestedIn` would return `rs=`{`('José','Databases')/1, ('Juan','Databases')/0.8, ('José','Cooking')/0.4, ('Cayetana','Conceptual Modeling')/0.7`}. In the case of similarity relations, the query would return default vales for those that are not explicitly stored. For example, `SELECT id, id FROM related` yields the `rs=`{ `(1,4)/0.8, (1,2)/0.1, (4, 1)/0.8, (2,1)/0.1, (2,4)/`**0.1**`, (4,2)/`**0.1**`, (1,1)/1, (2,2)/1 ...)` }, where reflexivity and symmetry have been used along with a minimum transitive default value.

The connection proxy preprocess the `SQL` sentence and extends it with the information in the metadata[6], changing the query by detecting in the original

[6] We used the `Zql` libraries in the current implementation

one the extended definitions. For example, the simple query `SELECT name FROM sporadic` includes the `sporadic` extended definition will be changed for the table in which it is stored, and its membership field will be implicitly retrieved. Therefore, the query will be internally converted to `SELECT name, sporadic-m FROM users`.

A `ResultSet` proxy is used to allow the retrieval of membership grades as shown in the following code fragment.

```
String query = "SELECT * FROM loyal WHERE points > 20
                   AND initialDate < '2002-11-03'";
ResultSet rs = stmt.executeQuery(query);
FuzzyResultSet frs = (FuzzyResultSet)rs;
while (rs.next()){
    System.out.println( "(" + rs.getString("username")+
            ","+ frs.getMembership() +")");
}
```

The query of the above fragment will obtain the tuples that match the two crisp conditions in the `WHERE` clause, yielding the users with `oid` 1 and 4. In addition, it will ask the `Loyal` class for their membership value, that will be computed from some of the (inherited or not) attributes of `loyal`. They will always be less or equal than 0.7 and 0.3, that are their respective membership grades in `premium`.

4 Some Usability and Performance Facts

The resulting interfaces are an strictly additive extension in the sense that they're fully compatible with the old interfaces, and the driver can be used with legacy code without modification. All the semantic assumptions of `JDBC` are preserved.

An experiment was carried out with four developers with more than a year of expertise in developing `JDBC`-based applications. They were provided with a 30 minutes introduction to the concepts of fuzzy relations and associations, and after that, they were asked to develop three code fragments, involving a single `SQL` sentence each one, using the `javadoc` documentation of `fJDBC` and a `UML` version of the database schema. All of them required sorting the `ResultSet` by membership value. They required an average of about 20 minutes to complete them, and the only problematic concept that they pointed out by two of them was that the use of a *T-norm* seemed them counterintuitive.

Regarding performance, the three approaches for fuzzy relations described in Section 2 differ significantly in their properties. The third one depends on the kind of queries that are needed to yield the membership value, and therefore can't be generally measured. For the other two approaches, Figure 7 shows the difference in retrieval time (in seconds) depending on the type of the implementation. In both cases, two hundred fuzzy relations (subsets) around the concept of *user* were defined [7], and stored in a `SQL Server 7` database on an isolated

[7] In the context of a personalized Web application, in which this amount of vague user groups may be found

network node (analogous results were obtained from a `DB2` database). The Figure shows the number of users in the X-axis, and retrieval time in seconds in the Y-axis. In the first configuration (left graphic), the membership of a specific user in all of the subsets was queried. The second configuration (the right graphic) shows retrieval times obtaining all the tuples that belong to a specific concept (set).

The first configuration shows that, as expected, implementing fuzzy sets as attributes in the same table (marked with triangles) is largely independent of the cardinality of the base relation, but implementing them as separate tables (marked with squares) makes performance fall even with moderate cardinalities (possible due to the loss of efficiency in the large chain of *join* operations). The inverse occurs in the second scenario (a possible reason for this behavior would be that the query needs to retrieve entire tuples, which carry a large amount of irrelevant information, i.e. the membership of each user in the other subgroups). As a conclusion, frameworks should left open to developers the implementation mode, since it impacts performance in different ways.

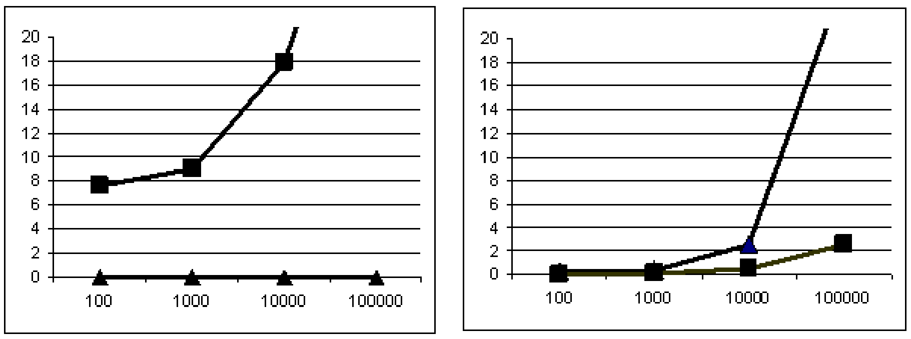


Fig. 7. Comparison of the two main approaches for fuzzy relations

Metadata processing does not impact performance significantly, since it's executed only once, due to the fact that drivers are designed as *Singletons* [9], as a consequence of the driver management mechanism. This mechanism entails that a single instance of each class implementing the `java.sql.Driver` interface must be passed to the `java.sql.DriverManager` class (via the `registerDriver` class method) for each application (i.e. for each independent `JVM`). That instance is used for every connection with the same protocol during the entire application lifetime[8].

[8] A different mechanism is provided as an alternative in `JDBC` 3.0 through the `javax.sql.DataSource` interface for distributed data sources, but it allows for a similar approach.

5 Conclusions and Future Work

`fJDBC` is an enhanced API that introduces three essential forms of logical fuzziness representation in existing relational databases, which can be used to implement three enhanced conceptual object-modelling constructs, and that exhibit different properties regarding performance and ease of extension from existing database schemas. Data access programming concepts only require a few added methods, and the `SQL` syntax remains the same, easing the adoption of the framework by the development community.
An analysis of the expressive power of `fJDBC` and its possible extensions (including fuzziness at the attribute level) will be carried out. Future work will address also the appropriateness of different scenarios, like multiple inheritance and *n-ary* associations in terms of efficiency and usability. In addition, a end-user interface for the definition of fuzzy relations and associations on existing database schemas is planned.

References

1. Baldwin, J.F., Zhou, S.Q.: A Fuzzy Relational Inference Language. Fuzzy Sets and Systems **14** (1984) 155–174
2. Berners-Lee, T., Fielding, R. Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. Internet Draft Standard, August, RFC2396 (1998)
3. Blaha, M., Premerlani, W.: Object-Oriented Modeling and Design for Database Applications. Prentice Hall, Upper Saddle River, New Jersey (1998)
4. Bosc, P., Pivert, O.: Fuzzy Querying in Conventional Databases, In: Zadeh, L., Kacprzyk, J. (eds.): Fuzzy Logic for the Management of Uncertainty. John Wiley, New York (1992) 645–671
5. Buckles, B.P., Petry, F.E: A Fuzzy Representation of Data for Relational Databases. Fuzzy Sets and Systems **7** (1982) 213–226
6. De Caluwe, R. (ed.): Fuzzy and Uncertain Object-Oriented Databases, Concepts and Models. World Scientific, Singapore (1997)
7. Chen, G.: Fuzzy Logic in Data Modelling: Semantics, Constraints, and Database Design. The Kluwer International Series on Advances in Database Systems, Kluwer (1998)
8. Galindo, J., Medina, J.M., Pons, O., Cubero, J.C.: A Server for Fuzzy SQL Queries. In: Andreasen, T., Christiansen, H., Larsen, H.L. (eds.): Lecture Notes in Artificial Intelligence (LNAI), Vol. 1495, Springer-Verlag, Berlin Heildelberg New York (1998) 164–174
9. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns. Elements of Reusable Object Oriented Design. Addison Wesley (1995)
10. Klir, G. J., Folger, T. A.: Fuzzy Sets, Uncertainty, and Information. Prentice Hall, Englewood Cliffs, New Jersey (1988)
11. Medina, J M. and Pons, O. and Vila, M A.: GEFRED. A Generalized Model of Fuzzy Relational Databases. Information Sciences **76**(1–2) (1994) 87–109
12. Object Management Group, OMG Unified Modeling Language Specification, Version 1.3. June 1999
13. Prade, H., Testemale, C.: Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries. Information Sciences **34** (1984) 115–143

14. Rasmussen, D., Yager, R.R.: SummarySQL – A Flexible Fuzzy Query Language. In: Proceedings of the 1996 Workshop on Flexible Query-Answering Systems (FQAS'96), Roskilde, Denmark, May 22–24 (1996) 1–18
15. Shenoi, S., Melton, A.: Proximity Relations in the Fuzzy Relational Database Model. Fuzzy Sets and Systems **31** (1989) 285–296
16. Sicilia, M.A., García, E., Gutiérrez-de-Mesa, J.A.: Integrating Fuzziness in Object–Oriented Modelling Languages: Towards a Fuzzy–UML. In: Klement, P.E., Mesiar, R., Drobná, E., Chovanec, F.: Proceedings of the sixth International Conference on Fuzzy Sets Theory and its Applications (FSTA'02), The Slovak Republic (2002) 66–67
17. Sicilia, M.A., Díaz, P., Aedo, I., García, E.: Fuziness in Adaptive Hypermedia Models. In: Proceedings of the 2002 North American Fuzzy Information Processing Society Conference (NAFIPS), Special Track 'Fuzzy Sets and the Internet', New Orleans, USA. IEEE Press (2002) 268–273
18. Umano, M.: Freedom-O A Fuzzy Database System. In: Gupta, M., Sánchez, E. (eds.): Fuzzy Information and Decision Processes. North-Holland, Amsterdam (1982) 339–347
19. World Wide Web Consortium (W3C): Extensible Markup Language (XML) 1.0. World Wide Web Consortium Recommendation. Available at http://www.w3.org/TR/REC-xml
20. Yazici, A., Cibiceli, D.: An Access Structure for Similarity–Based Fuzzy Databases. Information Sciences **115**(1–4) (1999) 137–163
21. Zadrozny, S., Kacprzyk, J: FQUERY for Access: Towards Human Consistent Querying User Interfaces. In: Proceedings of the 1996 ACM Symposium on Applied Computing (SAC'96) (1996) 532–536

Approximating Terminological Queries

Heiner Stuckenschmidt and Frank van Harmelen

Vrije Universiteit Amsterdam
de Boelelaan 1081a,
1081 HV Amsterdam, The Netherlands
{heiner,frankh}@cs.vu.nl

Abstract. Current proposals for languages to encode terminological knowledge in intelligent systems support logical reasoning for answering user queries about objects and classes. An application of these languages on the World Wide Web, however, is hampered by the limitations of logical reasoning in terms of efficiency and flexibility. In this paper we describe, how techniques from approximate reasoning can be used to overcome these problems. We discuss terminological knowledge and approximate reasoning in general and show the benefits of approximate reasoning using the example of building and maintaining semantic catalogues that can be used to query resource locations based on object classes.

1 Motivation

Recent research on the so-called Semantic Web aims at providing models and methods to make information on the World Wide Web accessible to machines instead of humans users. This aim requires a precise semantic description of information items, because different from human users, machines are not able to distinguish relevant from irrelevant in the absence of a precise description of its meaning. It has been argued that ontologies are a technology that provides such precise definitions of the terminology used in information sources [5]. Consequently much work has been done on the development of an infrastructure for representing and reasoning about ontologies on the web. Languages such as OIL (building on RDF Schema [1], later extended into DAML+OIL [13] and currently being standardized by W3C as OWL) used to encode terminological knowledge and advanced reasoning systems like RACER [7] exist that can be used to check and query terminological knowledge.

A strength of the current proposals for the foundational languages of the Semantic Web (RDF Schema, DAML+OIL), it that they are all based on formal logic. However, this reliance on logics is not only a strength but also a weakness. Traditionally, logic has always aimed at modeling idealized forms of reasoning under idealized circumstances. Clearly, this is not what is required under the practical circumstances of the Semantic Web. In a distributed and heterogeneous environment like the World Wide Web, we face a situation, where ontologies are built by many people that are not experts in knowledge modeling independent of each other. The resulting models will often lack precision or even contain contradictory information. These problems together with the intimidating size of the Web leads to the need for reasoning under time-pressure, reasoning with other limited resources besides time (e.g. limited memory, incomplete knowledge), reasoning that is not "perfect" but instead "good enough" for given tasks under given circumstances and algorithms that do not behave as yes/no oracles, but that instead display anytime behaviour [3]. It is tempting to conclude that symbolic, formal logic fails on all these counts, and to abandon that paradigm. However, research in the past few years has developed approximate reasoning methods with the above properties while staying within the framework of symbolic, formal logic [10,14,2,12].

In this paper we propose an approach for approximating the answers to terminological queries based on non-equivalent transformations of increasing exactness. We start with a general definition of terminological knowledge. We then discuss queries over terminological knowledge and review the approach of [9] for answering such queries. As the main contribution of the paper we describe different strategies for approximating answers to terminological queries by modifying the query answering approach mentioned above. We summarize with a discussion of open problems and further options for computing approximate answers.

2 Preliminarities: Terminological Knowledge

Before we can discuss the application of approximate reasoning techniques to terminological knowledge on the World Wide Web, we first have to clarify our view on terminological knowledge. We do this by defining terminological knowledge bases on a level concrete enough to precisely define reasoning tasks and abstract enough to be independent of a concrete language. These definitions are consistent with models of terminological knowledge based on Description Logics [4], which are underlying much of the current work on ontology languages for the Semantic Web such as OIL, DAML+OIL and OWL Based on this definition, we define a number of terminological queries, we might want to state and give a general notion of approximate results of such queries.

2.1 Knowledge Bases

A number of languages for encoding terminological knowledge on the Web have been proposed (see [6] for an overview). In order to get a general notion of terminological knowledge, we define the general structure of a terminological knowledge base independent of a concrete language.

Definition 1 (Terminological Knowledge Base). *A Terminological Knowledge Base $\mathcal{T}$ is a triple*

$$\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$$

where $\mathcal{C}$ is a set of class definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions.

Terminological knowledge usually groups objects of the world that have certain properties in common (e.g. cities or countries). A description of the shared properties is called a class definition. Concepts can be arranged into a subclass-superclass relation in order to be able to further discriminate objects into subgroups (e.g. capitals or European countries). Classes can be defined in two ways, by enumeration of its members or by stating that it is a refinement of a complex logical expressions. The specific logical operators to express such logical definitions can vary between ontology languages; the general definitions we give here abstract from these specific operators.

Definition 2 (Class Definitions). *A class definition is an axiom of one of the following forms:*

- $c \equiv (o_1, \cdots, o_n)$ *where c is a class definition and $o_1, \cdots, o_n$ are object definitions.*
- $c_1 \sqsubseteq c_2$ *where c_1 and c_2 are class definitions.*

Further, there is the universal class denoted as $\top$.

Objects of the same type normally occur in similar situations where they have a certain relation to each other (cities lie in countries, countries have a capital). These typical relations can often be specified in order to establish structures between classes. Terminological knowledge considers binary relations that can either be defined by restricting their domain and range or by declaring it to be a sub-relation of an existing one.

Definition 3 (Relation Definitions). *A relation definition is an axiom of one of the following forms:*

- $r \sqsubseteq (c_1, c_2)$ *where r is a role definition and c_1 and c_2 are class definitions.*
- $r_1 \sqsubseteq r_2$ *where r_1 and r_2 are role definitions.*

The universal role is defined as $\top \times \top$.

Sometimes single objects (e.g. the continent Europe) play a prominent role in a domain of interest, or the membership of a concept is defined by the relation to a specific object (European countries are those contained in Europe). For this purpose ontology languages often allow to specify single objects, also called instances. In our view on terminological knowledge, instances can be defined by stating their membership in a class. Further, we can define instances of binary relations by stating that two objects form such a pair.

Definition 4 (Object Definitions). *An object definition is an axiom of one of the following forms:*

- $o : c$ *where* c *is a class definition and* o *is an individual.*
- $(o_1, o_2) : r$ *where* r *is a relation definition and* o_1, o_2 *are object definitions.*

In the following, we will consider terminological knowledge bases that consist of such axioms. Of course, any specific ontology language will have to further instantiate these definitions to specify logical operators between classes etc, but for the purposes of this paper, these general definitions are sufficient. Further, we define the signature of a terminological knowledge base to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{IN} \rangle$, where $\mathcal{CN}$ is the set of all names of classes defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names and $\mathcal{IN}$ the set of all object names occurring in the knowledge base.

2.2 Semantics and Logical Consequence

We can define semantics and logical consequence of a terminological knowledge base using an interpretation mapping $.^{\Im}$ into an abstract domain Δ such that:

- $c^{\Im} \subseteq \Delta$ for all class definitions c in the way defined above
- $r^{\Im} \subseteq \Delta \times \Delta$ for all relation definition r
- $o^{\Im} \in \Delta$ for all object definitions o

This type of denotational semantics is inspired by description logics [4], however, we are not specific about operators that can be used to build class definitions which are of central interest of these logics. Using the interpretation mapping, we can define the notion of a model in the following way:

Definition 5 (Model of a Terminological Knowledge Base). *An interpretation* $\Im$ *is a model for the knowledge base* $\mathcal{T}$ *if* $\Im \models A$ *for every axiom* $A \in (\mathcal{C} \cup \mathcal{R} \cup \mathcal{O})$ *where* $\models$ *is defined as follows.*

- $\Im \models c \equiv (o_1, \cdots, o_n)$, *iff* $c^{\Im} = \{o_1^{\Im}, \cdots, o_n^{\Im}\}$
- $\Im \models c_1 \sqsubseteq c_2$, *iff* $c_1^{\Im} \subseteq c_2^{\Im}$
- $\Im \models r \sqsubseteq (c_1, c_2)$, *iff* $r^{\Im} \subseteq c_1^{\Im} \times c_2^{\Im}$
- $\Im \models r_1 \sqsubseteq r_2$, *iff* $r_1^{\Im} \subseteq r_2^{\Im}$
- $\Im \models o : c$, *iff* $o^{\Im} \in c^{\Im}$
- $\Im \models (o_1, o_2) : r$, *iff* $(o_1^{\Im}, o_2^{\Im}) \in r^{\Im}$

Having defined the notion of model, logical consequence can be defined in a straightforward way:

Definition 6 (Logical Consequence). *An axiom* A *logically follows from a set of axioms* $\mathcal{S}$ *if* $\Im \models \mathcal{S}$ *implies* $\Im \models A$ *for every model* $\Im$*. We denote this fact by* $\mathcal{S} \models A$*.*

Our work heavily relies on this notion of model and logical consequence, because query results are determined with respect to being the logical consequence of a certain set of axioms. Further, we need the notion of a model in order to characterize approximate results.

3 Querying Terminological Knowledge

In the following we first define ontology based queries as well as the notion of answers to and relations between queries. We formalize queries in the following way: conjuncts of a query are predicates that correspond to classes and relations of the ontology. Further, variables in a query may only be instantiated by constants that correspond to objects in that ontology.

Definition 7 (Terminological Queries). *Let V be a set of variables disjoint from $\mathcal{IN}$ then an terminological query Q over a knowledge base $\mathcal{T}$ is an expressions of the form*

$$Q \leftarrow q_{1_i} \wedge \cdots \wedge q_{m_i}$$

where q_i are query terms of the form $x : C$ or $(x, y) : R$ such that $x, y \in V \cup \mathcal{IN}$, $C \in \mathcal{CN}$ and $R \in \mathcal{RN}$.

We use the following toy example of a terminological knowledge base to illustrate their approach and the extensions we propose:

$$\begin{align} Human &\sqsubseteq \top \tag{1}\\ Male &\sqsubseteq Human \tag{2}\\ Female &\sqsubseteq Human \tag{3}\\ Organization &\sqsubseteq \top \tag{4}\\ University &\sqsubseteq Organization \tag{5}\\ Title &\sqsubseteq \top \tag{6}\\ father\text{-}of &\sqsubseteq Male \times Human \tag{7}\\ works\text{-}at &\sqsubseteq Human \times Organization \tag{8}\\ degree &\sqsubseteq Human \times Title \tag{9}\\ granted\text{-}by &\sqsubseteq Title \times University \tag{10}\\ vu &: University \tag{11} \end{align}$$

In particular, we consider the following query formulated over the example knowledge base:

$$\begin{align} Q(X) \leftarrow\ & (X, Y) : father\text{-}of \wedge (Y, Z) : works\text{-}at \wedge Y : Female \wedge \\ & (Y, V) : degree \wedge (V, vu) : awarded\text{-}by \tag{12} \end{align}$$

This query asks for all persons that are father of a female person that has a degree awarded by the Vrije Universiteit Amsterdam and works somewhere.

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to terminological queries in terms of instantiations of the query that are logical consequences of the knowledge base it refers to:

Definition 8 (Query Answers). *The answer of a query Q containing variables $v_1, \cdots, v_k$ over a knowledge base T is a set of tuples $(i_1, \cdots, i_k)$ such that $T \models Q'$ where Q' is the query obtained from Q by substituting $v_1, \cdots, v_k$ by $i_1, \cdots, i_k$. The projections of the answer tuples to variables occurring in the head of a query Q is denoted as $res(Q)$ and referred to as the answer set of Q.*

Based on the possible answers to a query, we can define semantic relations between different queries we will later use to characterize approximations. In particular, we consider query containment and query equivalence (compare [8])

Definition 9 (Query Containment and Equivalence). *Let $\mathcal{T} = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ and Q_1, Q_2 conjunctive queries over $\mathcal{T}$. Q_1 is said to be contained in another query Q_2 denoted by $Q_1 \sqsubseteq Q_2$ if for all possible sets of object definitions of a terminological knowledge base the answers for Q_1 is a subset of the answers for Q_2 : $(\forall \mathcal{O} : res(Q_1) \subseteq res(Q_2))$. The two queries are said to be equivalent, denoted as $Q_1 \equiv Q_2$ iff $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$*

An example for query subsumption is the following query that can be shown to subsume the example query above:

$$\begin{aligned} Q'(X) \leftarrow X : Male \wedge (X,Y) : father\text{-}of \wedge Y : Female \wedge \\ (Y,Z) : degree \wedge (Z,U) : awarded\text{-}by \wedge U : University \end{aligned} \quad (13)$$

There are three points where this query differs from Q. First of all the awarding organization is not restricted to a specific one, but only to universities in general. Further, the conjunct concerning the *works-at* relation is missing. For these two changes, it is easy to see that the set on possible answers includes all the answers of Q. The last difference, which is the additional information that X should be of type male seems to be more restrictive than the corresponding restriction in Q. However, as the $father\text{-}of$ relation is defined over the domain of male persons, this additional conjunct does not change the set of possible answers.

Based on these semantic relation between queries, we can distinguish three different kinds of transformations, namely equivalent transformations resulting in a query that is equivalent to the original query, contained and containing transformations.

Definition 10 (Query Transformations). *Let Q and Q' be queries over the same Knowledge base, then*

- *A query Q' is an equivalent transformation of Q if $Q \equiv Q'$.*
- *A query Q' is a contained transformation of Q if $Q \sqsubseteq Q'$*
- *A query Q' is a containing transformation of Q if $Q' \sqsubseteq Q$*

In the following section, we discuss a method for actually computing the result of a conjunctive query over a terminology. We restrict ourselves to queries with only one variable in the query head.

4 Approximating Conjunctive Queries

The underlying assumption of our approach is that less complex queries can be answered in shorter time. Following this idea, we propose to compute a sequence $Q^1, \cdots, Q^n$ of queries starting with very simple ones while gradually increasing their complexity. In order to use these simpler queries as approximations for the original query we have to ensure the following properties of the sequence of queries:

1. $i < j \implies Q^i \sqsupseteq Q^j$
2. $Q_n \equiv Q$

The first property ensures that the quality of the results of the queries is not decreasing. The second claims that the last query computed returns the desired exact result. Together, these properties ensure that the sequence of queries approximate the exact result.

Next, we have to define how to determine queries to be used for the approximation. This process is constrained by the two properties above. We decide to start with the universal query Q^0 that returns all objects in the knowledge base and to successively add conjuncts from the original query. It is easy to see that this gives us the desired properties as adding a conjunct leads to a subsumed query. Further, as the original query has a finite number of conjuncts, adding conjuncts necessarily leads to the original query after a finite number of steps. The critical part concerning this approach is the step of deciding which conjunct(s) to add next in order to generate the next query in the sequence. This choice has a strong influence on the quality of the approximation as a badly chosen ordering may lead to a situation, were the answer set remains the same for a long time and only reduces to the exact answer in the last step. In the following, we discuss different strategies for determining sequences of subsuming queries that try to avoid the problem mentioned.

4.1 Node Expansion Approximation

An approximation that actually increases the quality of the result in every step is considered a good approximation. In order to achieve such an approximation, we have to try to specify the next query in the sequence in such a way that it excludes objects from the current answer set. In order to produce good approximations and not just arbitrary ones, we have to make sure that the next conjuncts added to the query expression directly apply to the objects in the answer set. This in turn depends on the dependencies between the variables in the query. In order to keep track of such dependencies Horrocks and Tessaris [9] introduce the notion of a query graph.

Definition 11 (Query Graph (Horrocks and Tessaris 2000)). *The graph induced by a query is a directed graph with a node for every variable and individual name in the query and an directed edge from node x to node y for every role term $(x, y) : R$ in the query.*

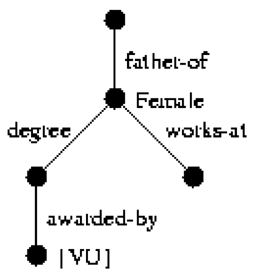


Fig. 1. Query Graph of the Example Query

While the approach of Horrocks and Tessaris is more general, we restrict ourselves to queries where the query graph is a (directed) tree and its root node corresponds to the variable we are interested in. In particular, this requires that none of the roles used in the query is declared to be functional and that each constant only appears once in a query. We can use the query graph as a basic structure to determine the next query in the sequence. In especially, we can start with the query that corresponds to the node of the query graph representing the answer variable. Starting from this node, we then successively expand nodes by adding their children and the corresponding arcs. Figure 2 shows the query graphs corresponding to the sequence of queries we obtain for the example query by applying the node expansion strategy.

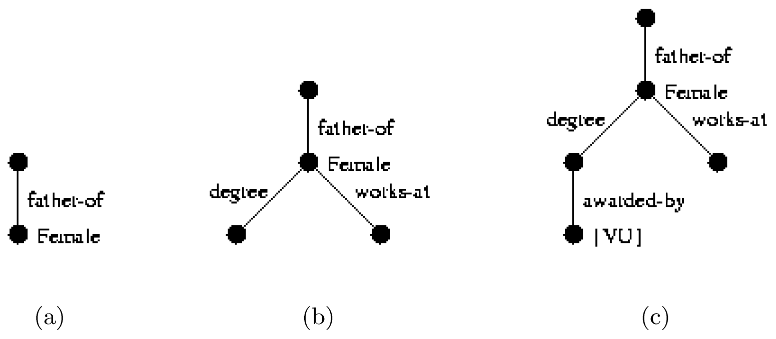


Fig. 2. Sequence of Query-Graphs for Node Expansion Approximation

The intuition of this approach is that in every step, we select one variable from the query and apply add all restrictions from the original query that apply to this specific variable. Applying these restrictions makes it very likely that the set of objects that can be used to instantiate this variable is reduced. The structure of the query graph ensures that this restriction has a potential influence on the set of objects we are interested in. Therefore it is also likely that the set of possible answers to the query is reduced.

We can also relate this back to the original idea of successively adding conjuncts to the universal query until we get the original query. The sequence of queries described above relates to the following series of conjunctive queries. We replace re-occurring conjuncts form previous queries in the sequence by the name of the corresponding query for better readability.

- $Q^1(X) \leftarrow (X,Y) : father\text{-}of \wedge Y : Female$
- $Q^2(X) \leftarrow Q^1(X) \wedge (Y,Z) : works\text{-}at \wedge (Y,V) : degree$
- $Q^3(X) \leftarrow Q^2(X)(V,vu) : awarded\text{-}by$

We can easily see that this sequence of queries satisfies the requirements as they are strict extensions of each other and Q^3 is equal to Q.

4.2 Arc-Based Approximations

A potential problem of the node expansion strategy presented above is the granularity of the modifications made in each step of the approximation process. The approach of adding all restrictions on a variable at once may be problematic, because the change in the answer set may be radical, especially if many restrictions to a single variable are contained in the query. Adding many restriction at once will also often lead to a very small number of possible approximations. in fact, in this case the number of approximations is equal to the number of distinct variables in the query. In general, however, the number of possible approximations is equal to the number of possible subtrees of the original query trees. In order to overcome this problem, we now consider approximation strategies that ensure that only a limited number of restrictions are added at a time. This is achieved by only expanding the query graph one arc at a time. Using this approach, choosing the next expansion step is even more difficult, because we do not only have to decide which node to expand, but also which arc to add to that node. In the following we introduce two general strategies for selecting arcs to expand based on well-known search techniques.

Breadth-First Approximation: The first possibility to apply arc-based approximations is to adopt a breath-first strategy where first all arcs on the highest level of the tree are expanded one at a time before moving to the next level. The corresponding sequence of query graphs is shown in figure 3.

The intuition of this strategy is to first restrict variables that are closely related to the goal variable. The assumption is that the instantiation of these variables have a stronger influence on the objects in the query set.

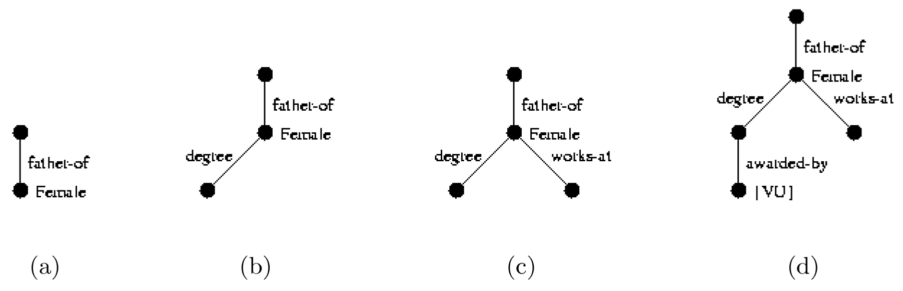


Fig. 3. Sequence of Query-Graphs for Breath-First Approximation

Looking at the corresponding conjunctive queries, we see that only one conjunct containing a binary predicate and therefore restricting the property of an object is added at each step of the approximation.

- $Q^1(X) \leftarrow (X,Y) : father\text{-}of \wedge Y : Female$
- $Q^2(X) \leftarrow Q^1(X) \wedge (Y,V) : degree$
- $Q^3(X) \leftarrow Q^2(X) \wedge (Y,Z) : works\text{-}at$
- $Q^4(X) \leftarrow Q^3(X) \wedge (V,vu) : awarded\text{-}by$

Depth-First Approximation: Analogously, we can apply a depths-first strategy for determining the next expansion of the query tree, In this case nodes at the deepest level of the partially expanded tree are expanded first. After the deepest level is reached, the expansion is continued at the second lowest level of the same branch and so on. The corresponding sequence of query trees is shown in figure 4.

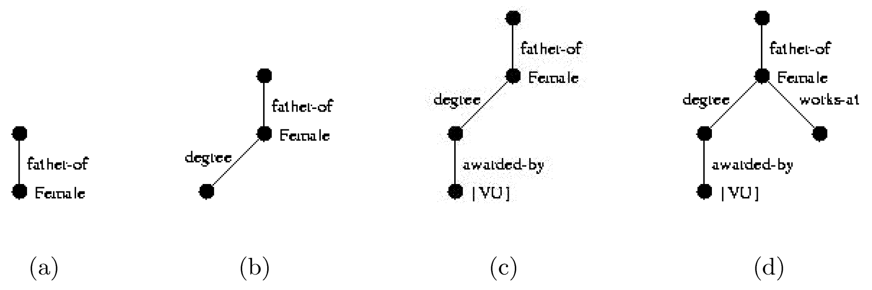


Fig. 4. Sequence of Query-Graphs for Depth-First Approximation

The intuition behind this rather strange looking strategy is to give a preference to role chains occurring the data. This choice is motivated by the assumption that long role chains are rather an exception and therefore provide a strong criterion for excluding objects from the answer set early in the approximation process.

In the corresponding sequence of conjunctive queries we can observe a preference for first introducing new variables before further restricting the old ones. As a consequence, a larger number of variables is restricted early in the approximation process. This also makes a significant reduction of possible answers in the early stages of the approximation likely.

- $Q^1(X) \leftarrow (X,Y) : \mathit{father\text{-}of} \land Y : \mathit{Female}$
- $Q^2(X) \leftarrow Q^1(X) \land (Y,V) : \mathit{degree}$
- $Q^3(X) \leftarrow Q^2(X) \land (V,vu) : \mathit{awarded\text{-}by}$
- $Q^4(X) \leftarrow Q^3(X) \land (Y,Z) : \mathit{works\text{-}at}$

5 Query Execution

Actually executing conjunctive queries over terminologies is a difficult task because unlike conjunctive queries over a database, there is not a single minimal model that satisfies the query. Due to this fact, query answering in the face of complex terminologies needs deductive reasoning. In the following, we describe the approach for answering conjunctive queries over terminologies proposed by Horrocks and Tessaris [9]. The idea of the approach of Horrocks and Tessaris is to translate the query into an equivalent concept expression, classify this new concept and use standard inference methods to check whether an object is an instance of the concept corresponding to the query expression. This approach makes use of the fact that binary relations in a conjunctive query can be translated into an existential restriction in such a way that logical consequence is preserved after a minor modification of the A-Box. Details are given in the following theorem.

Theorem 1 (Role Roll-Up (Horrocks and Tessaris 2000)). *Let $\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ be a terminological knowledge base with T-Box T and A-Box A. Let furtherR be a role, C_I Concept names in T and a, b be individual names in A. Given a new concept name P_b not appearing in T, then*

$$\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle \models (a,b) : R \land b : C_1 \land \cdots \land b : C_k$$

if and only if

$$\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \cup \{b : P_b\} \rangle \models a : \exists R(P_b \sqcap C_1 \sqcap \cdots \sqcap C_k)$$

The theorem directly applies to parts of our query. Consider for example the two conjuncts $(X,Y) : \mathit{father\text{-}of}$ and $Y : \mathit{Female}$. From the theorem we get that the instances of the concept $(\exists \mathit{father\text{-}of}.\mathit{Female})$ are exactly the objects to instantiate X we were looking for. The introduction of a new primitive concept is not necessary as we are not checking whether a concrete object satisfies the query expressions. The transformation of a complete query is more difficult due to the dependencies between the variables that occur in the query expression. The correct transformation of a query into a concept expression depends on the kinds of dependencies between the variables in the query which is reflected in the structure of the query graph. Using the information provided by the query graph, the transformation of a conjunctive query into a concept expression, called roll-up, can be computed as follows:

Definition 12 (Query Roll-up (Horrocks and Tessaris 2000)). *the roll-up of a query Q with query tree G is a concept expression derived from Q by successively applying the following rule:*

- *If G contains a leaf node y then the role term (x,y):R is rolled up according to theorem 1. The edge (x,y) is removed from G*

Using this roll-up mechanism, we can translate our example query into the concept expression shown in the equation below. The results of Horrocks and Tessaris guarantee that every answer to the query is a member of this concept and vice versa.

$$(\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$$

Using the query graph as a basis for determining the sequence of queries has the advantage that we already obtain the information needed for transforming the query into a concept expression being used to actually answer the query. If we apply the role-up algorithm of Horrocks and Tessaris to the query graphs in figure 2 we get the following sequence of concept expressions defining supersets of the set of answers we are looking for:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top) \sqcap (\exists works\text{-}at.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

In the same way we get the following sequence of concept expressions for the breaths-first strategy:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top) \sqcap (\exists works\text{-}at.\top)))$
Q^4: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

For the depths-first strategy, the following sequence is generated:

Q^1: $\top \sqcap (\exists father\text{-}of.Female)$
Q^2: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.\top)))$
Q^3: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\}))))$
Q^4: $\top \sqcap (\exists father\text{-}of.(Female \sqcap (\exists degree.(\exists awarded\text{-}by.\{vu\})) \sqcap (\exists works\text{-}at.\top)))$

The role of this mechanism is two-fold. As Horrocks and Tessaris show, it can be used to actually execute queries using existing subsumption reasoners. Further, using a description logic reasoner, we can easily prove that the two requirements on the sequence of queries stated in the intriduction of section 4 actually hold. In the cases discussed here, it is straightforward to see that the requirements hold as we are just adding conjuncts ending up with exactly the same query we want to approximate. Using a subsumption reasoner, however, enables us to check the requirements for arbitrary sequences of query.

6 Discussion

Querying terminological knowledge bases is a problem that currently gains a lot of importance as it can be assumed to be one of the main inference tasks on the semantic web. Existing proposals for query answering approaches are able to answer queries over very complex knowledge bases but at the price of a high complexity resulting from the need to perform deductive reasoning over the knowledge base. In order to improve the efficiency of this approach, we propose various strategies for approximating answers by relaxing the query expression to a subsuming one that still returns all correct answers but possibly also some wrong ones. As the main application we have in mind is the search for information on the web, returning some wrong answers is acceptable as the result is returned to the user for validation. Further, our approximation strategies is designed in such a way that the exact result is also computed if there is enough time. These characteristics make our approach a promising one for gaining efficiency in semantic web querying despite the fact that it is very premature. Possible further improvements are the following:

Approximation Heuristics: The different strategies presented in this paper were solely based on the structure of the query graph. The different approximation steps were motivated by uninformed search strategies. However, it has been shown in other domains such as diagnosis or configuration that approximation strategies can be significantly improved using knowledge about the problem domain [12]. In our case, this knowledge is provided in terms of the concept and role definitions in the terminological knowledge base as well as in terms of concepts and roles occurring in the approximated query. An interesting question for further research is whether we can use this domain knowledge to define informed approximation strategies that are not solely guided by the structure of the query graph, but also by the nature of the concepts and roles. In particular, such knowledge can be used to determine the order, in which links are used to expand the query graph. We might want to add those conjuncts first that are most restrictive, thereby giving a better approximation of the actual answer. Such a heuristic ordering could be applied in combination with one of the described strategies or as a completely heuristic strategy that always uses the heuristics to expand the tree.

Integrating Subsumption Reasoning: A second interesting extension of the proposed approach is to use the roll-up mechanism proposed by Horrocks and Tessaris in order to interleave query formulation with subsumption reasoning. The advantage of such an effort is two-fold. First of all, we get a way for checking query subsumption, doing this is straightforward for the approaches we discuss in this paper, but maybe there are more sophisticated strategies that lead to situations where query subsumption is not straightforward. Indeed using a subsumption reasoner to check query subsumption may become an essential part of the query generation process. Secondly, the rolled-up query is the way to actually execute the query on a complex knowledge base and to return results. These results (e.g. the properties of objects in the answer set) may also give hints towards ways to formulate the next query in the sequence. The choice of the next conjunct can then be based on its ability to remove objects from the previous answer set. A possible drawback of interleaving query formulation with subsumption reasoning is the computational complexity of subsumption checking having a negative impact on the overall performance of the approach. This is not so much a problem for the case of query execution as this inference step has to be done anyway. Testing query subsumption however is an additional reasoning task that should be avoided if possible.

Approximate Deduction: Up to now, we only considered approximations that are done in the transformation step of the query answering approach by Horrocks and Tessaris. Another option not being discussed here is to apply approximation techniques to the second step of the approach, namely the deduction step that actually computes the answer to the query based on the concept expression generated by the transformation. Existing approaches to this problem are known as approximate deduction. Examples of techniques for approximate deduction are the work of Cadoli and Schaerf [10] and of Dalal [2]. These techniques for approximate deduction are very general. They are typically dependent on the choice of specific parameters that determine the degree of approximation (e.g. the length of clauses [2] or a subset of the alphabet [10]). In general, it is hard to determine which values for these parameters give the best approximations. Approximate deduction approaches for terminological knowledge, which is relevant for our work have been developed by Selman and Kautz [11] as well as Schaerf and Cadoli [10]. The latter discuss approximations for specific logics that can be used to express the kind of concepts we get by transforming conjunctive queries which makes the work a good starting point for investigating the use of approximate deduction for query answering.

References

1. D. Brickley, R. Guha, and A. Layman. Resource description framework (rdf) schema specification. Working draft, W3C, August 1998. http://www.w3c.org/TR/WD-rdf-schema.
2. Mukesh Dalal. Semantics of an anytime family of reasoners. In *European Conference on Artificial Intelligence*, pages 360–364, 1996.

3. T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of AAAI-88*, pages 49–54, 1988.
4. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, , and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.
5. D. Fensel. *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*. Springer-Verlag, Berlin, 2001.
6. A. Gomez-Perez and O. Corcho. Ontology langauges for the semantic web. *IEEE Intelligent Systems*, January/February:54–60, 2002.
7. Volker Haarslev and Ralf Moller. Description of the RACER system and its applications. In *Proceedings of the Description Logics Worlshop DL-2001*, Stanford, CA, 2001.
8. Alon Y. Halevy. Theory of answering queries using views. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(4):40–47, 2000.
9. Ian Horrocks and Sergio Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.
10. Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74(2):249–310, 1995.
11. B. Selman and H. Kautz. Knowledge compilation and theory approximation. *Journal of the ACM*, 43(2):193–224, March 1996.
12. A. ten Teije and F. van Harmelen. Exploiting domain knowledge for approximate diagnosis. In M. Pollack, editor, *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 454–459, Nagoya, Japan, August 1997.
13. Frank van Harmelen, Peter F. Patel-Schneider, and Ian Horrocks. Reference description of the DAML+OIL (march 2001) ontology markup language. http://www.daml.org/2001/03/reference.html, march 2001.
14. S. Zilberstein and S.J. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*. Kluwer Academic Publishers, 1995.

Conjunctive Aggregation of Extended Possibilistic Truth Values and Flexible Database Querying

Guy de Tré, Rita de Caluwe, Jörg Verstraete, and Axel Hallez

Computer Science Laboratory, Department of Telecommunications and Information Processing, Ghent University, Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium
Guy.DeTre@telin.rug.ac.be

Abstract. Extended possibilistic truth values are a flexible means to model the linguistic (un)certainty about the truth value of a proposition. With respect to flexible database querying, they are suited to express the extent to which a database instance satisfies a query condition. Since a query can impose several conditions, aggregation is necessary. In this paper, three definitions of conjunctive aggregation operators for extended possibilistic truth values are presented and compared with each other on the basis of their capability to rank the possible alternatives in the result of a (flexible) database query.

Keywords: Flexible database querying, extended possibilistic truth values, conjunctive aggregation.

1 Introduction

An important topic in research on flexible database (querying and answering) systems is the modelling of constraint satisfaction. Constraints can be used to compel the semantics and integrity of the stored information or to express some querying criteria [1,2]. The 'degrees' to which a database instance satisfies these constraints determine the extent to which the instance belongs to the database or to the answer set of a query.

The fuzzy set theory and the related possibility theory provide adequate means to model constraint satisfaction. The current state of the art can be summarized in three approaches. The first approach is to associate a fuzzy grade of membership with each database object. This grade may be interpreted as a degree of satisfaction or as a degree of confidence [3]. Such an approach is taken, for instance, by Baldwin and Zhou [4]. The second approach is taken by Van Schooten [5], Van Gyseghem [6] and de Tré et al. [1], who associate a possibilistic truth value with each database object. This approach has been extended by using extended possibilistic truth values, which have been obtained from the additional assumption that constraint satisfaction can be undefined [7] (if the constraint is inapplicable). The third approach is based on the use of a $[0,1]$-valued possibility measure (and an associated necessity measure) [8,9] in order to represent (un)certainty. This approach is taken, for instance, by Prade and Testemale [10] and by Bordogna et al. [11].

This paper deals with the extension of the second approach. More specifically, the focus is on the conjunctive aggregation of extended possibilistic truth values in order to be able to handle querying criteria which impose more than one condition. Three definitions are presented and compared with each other on the basis of their applicability in flexible database querying. The comparison criterium being applied is the 'capability of the operator to rank the possible alternatives in the result of a query'. A suitable aggregation operator provides a ranking which matches intuition and which allows to distinguish adequately between the different instances of the result. Such an operator is necessary for a satisfactory representation of the results of a flexible database query.

The structure of the paper is the following: the concept 'extended possibilistic truth value' is presented in Section 2. In Section 3, three definitions for a conjunctive aggregation operator for extended possibilistic truth values are introduced. In Section 4, the presented definitions are compared with each other. Finally, the achieved results are summarized in the concluding Section 5.

2 Extended Possibilistic Truth Values

The concept 'extended possibilistic truth value' is defined as an extension of the concept 'possibilistic truth value' which was originally introduced by Prade in [12]. Possibilistic truth values provide an epistemological representation of the truth of a proposition, which allows to reflect our knowledge about the actual truth. Their semantics is defined in terms of a possibility distribution [12].

Definition 1 (Possibilistic Truth Value). *With the understanding that P represents the universe of all propositions and $\tilde{\wp}(I)$ denotes the set of all ordinary fuzzy sets that can be defined over the universal set $I = \{T, F\}$ (where T represents 'True' and F represents 'False'), the so-called possibilistic truth value $\tilde{t}(p)$ of a proposition $p \in P$ is formally defined by means of a mapping*

$$\tilde{t} : P \rightarrow \tilde{\wp}(I)$$

which associates with each $p \in P$ a fuzzy set $\tilde{t}(p)$. The semantics of the associated fuzzy set $\tilde{t}(p)$ is defined in terms of a possibility distribution. With the understanding that

$$t : P \rightarrow I$$

is the mapping function which associates the value T with p if p is true and associates the value F with p otherwise, this means that

$$(\forall\, x \in I)(\Pi_{t(p)}(x) = \mu_{\tilde{t}(p)}(x)),\ i.e.\ (\forall\, p \in P)(\Pi_{t(p)} = \tilde{t}(p))$$

where $\Pi_{t(p)}(x)$ denotes the possibility that the value of $t(p)$ conforms to x and $\mu_{\tilde{t}(p)}(x)$ is the membership grade of x within the fuzzy set $\tilde{t}(p)$.

The truth value of a proposition can be unknown. This is the case if some data in the proposition exist but are not available. For example, the truth value of the proposition "$a \ div \ 2 = 3$" is unknown if no information about the value of a is given. The unknown possibilistic truth value is modelled as $\{(T, 1), (F, 1)\}$.

For the definition of an extended possibilistic truth value, an extra element $\perp$, which represents an undefined truth value, is added to the universal set I [13]. The addition of the element $\perp$ is inspired by the assumption that the truth value of a proposition can be undefined. This is the case if the proposition cannot be evaluated due to the non-applicability of (some of) its elements. For example, the truth value of the proposition "$a \ div \ 0 = 3$" is undefined because division by zero is not defined.

Definition 2 (Extended Possibilistic Truth Value). *With the understanding that P represents the universe of all propositions and $\tilde{\wp}(I^*)$ denotes the set of all ordinary fuzzy sets that can be defined over the universal set $I^* = \{T, F, \perp\}$, the so-called extended possibilistic truth value $\tilde{t}^*(p)$ of a proposition $p \in P$ is formally defined by means of a mapping*

$$\tilde{t}^* : P \to \tilde{\wp}(I^*)$$

which associates with each $p \in P$ a fuzzy set $\tilde{t}^*(p)$. The semantics of the associated fuzzy set $\tilde{t}^*(p)$ is also defined in terms of a possibility distribution. With the understanding that

$$t^* : P \to I^*$$

is the mapping function which associates the value T with p if p is true, associates the value F with p if p is false and associates the value $\perp$ with p if (some of) the elements of p are not applicable, undefined or not supplied, this means that

$$(\forall \, x \in I^*)(\Pi_{t^*(p)}(x) = \mu_{\tilde{t}^*(p)}(x)), \textit{ i.e. } (\forall \, p \in P)(\Pi_{t^*(p)} = \tilde{t}^*(p))$$

where $\Pi_{t^*(p)}(x)$ denotes the possibility that the value of $t^*(p)$ conforms to x and $\mu_{\tilde{t}^*(p)}(x)$ is the membership grade of x within the fuzzy set $\tilde{t}^*(p)$.

Special cases are:

$\tilde{t}^*(p)$	interpretation
$\{(T, 1)\}$	p is true
$\{(F, 1)\}$	p is false
$\{(T, 1), (F, 1)\}$	p is unknown
$\{(\perp, 1)\}$	p is undefined
$\{(T, 1), (F, 1), (\perp, 1)\}$	p is unknown or undefined

New propositions can be constructed from existing propositions, using so-called logical operators. An unary operator '$\tilde{\neg}$' is provided for the negation of a proposition and binary operators '$\tilde{\wedge}$', '$\tilde{\vee}$', '$\tilde{\Rightarrow}$' and '$\tilde{\Leftrightarrow}$' are respectively provided for the conjunction, disjunction, implication and equivalence of propositions. The

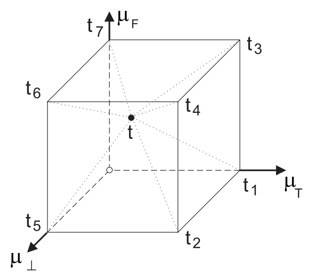


Fig. 1. The 3D-space of extended possibilistic truth values.

arithmetic rules to calculate the extended possibilistic truth value of a composite proposition and the algebraic properties of extended possibilistic truth values are presented in [13].

It turns out that the algebraic structure $(\tilde{\wp}(I^*), \tilde{\wedge}, \tilde{\vee}, \tilde{\neg})$ is not a lattice, so that there does not exist a natural partial ordering relation for this structure [13]. In order to have an idea of the relative importance of extended possibilistic truth values, a ranking function

$$r : \tilde{\wp}(I^*) \to [0, 1]$$

is introduced and obtained by considering the 3D-space of the extended possibilistic truth values as illustrated in Figure 1 and by associating a 'weight' w_i, $i = 1, 2, \ldots, 7$ with each relevant angular point t_i of the cube (the origin is excluded since it does not represent a valid truth value). For a given truth value $t = \{(T, \mu_T), (F, \mu_F), (\perp, \mu_\perp)\}$, $r(t)$ is defined as the 'weighted' sum of the Euclidian distances $d(t, t_i)$, $i = 1, 2, \ldots, 7$, i.e.

$$r(t) = \sum_{i=1}^{7} w_i \, d(t, t_i)$$

The 'weights' w_i, $i = 1, 2, \ldots, 7$ are calculated by considering the intuitive[1] ranking $t_1 > t_2 > t_3 > t_4 > t_5 > t_6 > t_7$ and by solving the linear system of equations $r(t_1) = 1$ (the best truth value), $r(t_2) = 5/6$, $r(t_3) = 4/6$, $r(t_4) = 3/6$, $r(t_5) = 2/6$, $r(t_6) = 1/6$ and $r(t_7) = 0$ (the worst truth value), i.e.

[1] The relevant angular points of the cube respectively have the following interpretations: $t_1 = \{(T, 1)\}$, $t_2 = \{(T, 1), (\perp, 1)\}$, $t_3 = \{(T, 1), (F, 1)\}$, $t_4 = \{(T, 1), (F, 1), (\perp, 1)\}$, $t_5 = \{(\perp, 1)\}$, $t_6 = \{(F, 1), (\perp, 1)\}$ and $t_7 = \{(F, 1)\}$. With respect to the modelling of truth, t_1 which represents 'true' is considered to be the best truth value, whereas t_7 which represents 'false' is considered to be the worst one. Taking into account these two opposite cases, the intuitive ranking $t_1 > t_2 > t_3 > t_4 > t_5 > t_6 > t_7$ can be justified.

$$\begin{cases} w_2 + w_3 + \sqrt{2}w_4 + \sqrt{2}w_5 + \sqrt{3}w_6 + \sqrt{2}w_7 = 1 \\ w_1 + \sqrt{2}w_3 + w_4 + w_5 + \sqrt{2}w_6 + \sqrt{3}w_7 = 5/6 \\ w_1 + \sqrt{2}w_2 + w_4 + \sqrt{3}w_5 + \sqrt{2}w_6 + w_7 = 4/6 \\ \sqrt{2}w_1 + w_2 + w_3 + \sqrt{2}w_5 + w_6 + \sqrt{2}w_7 = 3/6 \\ \sqrt{2}w_1 + w_2 + \sqrt{3}w_3 + \sqrt{2}w_4 + w_6 + \sqrt{2}w_7 = 2/6 \\ \sqrt{3}w_1 + \sqrt{2}w_2 + \sqrt{2}w_3 + w_4 + w_5 + w_7 = 1/6 \\ \sqrt{2}w_1 + \sqrt{3}w_2 + w_3 + \sqrt{2}w_4 + \sqrt{2}w_5 + w_6 = 0 \end{cases}$$

which yields $w_1 = -0.199$, $w_2 = -0.082$, $w_3 = -0.088$, $w_4 = 0.123$, $w_5 = 0.195$, $w_6 = 0.062$ and $w_7 = 0.434$.

The ranking function r has been used for the comparison of the presented aggregation operators.

3 Conjunctive Aggregation

Three definitions for a conjunctive aggregation operator for extended possibilistic truth values are presented. The first definition is an extension of the classical logical conjunction operator and is obtained by applying Zadeh's extension principle to Kleene's conjunction operator [14]. The second definition is based on the logical t-norm operation, whereas the third definition is based on the algebraic t-norm operation.

3.1 Extension of the Classical Logical Conjunction Operator

Zadeh's extension principle for fuzzy sets is in essence a basic identity which allows the arguments of the definition of an operator to be extended from elements in U to fuzzy sets in $\tilde{\wp}(U)$ [15] and can be meaningfully applied to the logical conjunction operator '$\wedge$' as defined by Kleene [14]. This approach is similar to the one presented by Prade in [12].

Definition 3 (Operator $\tilde{\wedge}$). *The 'extended' conjunction operator*

$$\tilde{\wedge} : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \,\tilde{\wedge}\, \tilde{V}$$

is defined by:

- $\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(T) = \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$
- $\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(F) = \max\begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(F)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(T)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F)), \\ \min(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(\bot)), \\ \min(\mu_{\tilde{U}}(\bot), \mu_{\tilde{V}}(F)) \end{pmatrix}$
- $\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(\bot) = \max\begin{pmatrix} \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(\bot)), \\ \min(\mu_{\tilde{U}}(\bot), \mu_{\tilde{V}}(T)), \\ \min(\mu_{\tilde{U}}(\bot), \mu_{\tilde{V}}(\bot)) \end{pmatrix}$

The operator $\tilde{\wedge}$ is "almost compositional" [17] since with $p, q \in P$, $\tilde{t}^*(p \wedge q) = \tilde{t}^*(p) \,\tilde{\wedge}\, \tilde{t}^*(q)$ always hold, except when $q = \neg p$.

3.2 Definitions Based on a t-(Co)norm Function

As alternatives for Definition 3, conjunctive aggregation operators which are based on a t-norm function and on a t-conorm function are proposed. Each t-norm function

$$i : [0,1] \times [0,1] \to [0,1] : (a,b) \mapsto i(a,b)$$

has to satisfy the following axioms [16]:

i1. $\forall\, a \in [0,1] : i(a,1) = a$
i2. $\forall\, a,b,c \in [0,1] : b \leq c \Rightarrow i(a,b) \leq i(a,c)$
i3. $\forall\, a,b \in [0,1] : i(a,b) = i(b,a)$
i4. $\forall\, a,b,c \in [0,1] : i(a,i(b,c)) = i(i(a,b),c)$

Analogously, each t-conorm function

$$u : [0,1] \times [0,1] \to [0,1] : (a,b) \mapsto u(a,b)$$

has to satisfy the following axioms [16]:

u1. $\forall\, a \in [0,1] : u(a,0) = a$
u2. $\forall\, a,b,c \in [0,1] : b \leq c \Rightarrow u(a,b) \leq u(a,c)$
u3. $\forall\, a,b \in [0,1] : u(a,b) = u(b,a)$
u4. $\forall\, a,b,c \in [0,1] : u(a,u(b,c)) = u(u(a,b),c)$

Definition 4 (Operators $\tilde{\wedge}_i$). *The alternative conjunctive aggregation operators*

$$\tilde{\wedge}_i : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \to \tilde{\wp}(I^*) : (\tilde{U},\tilde{V}) \mapsto \tilde{U} \,\tilde{\wedge}_i\, \tilde{V}$$

are defined by:

- $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(T) = i(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$
- $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(F) = u(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))$
- $\mu_{\tilde{U}\tilde{\wedge}\tilde{V}}(\perp) = u(u(i(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(\perp)), i(\mu_{\tilde{U}}(\perp), \mu_{\tilde{V}}(T))), i(\mu_{\tilde{U}}(\perp), \mu_{\tilde{V}}(\perp)))$

where i is a t-norm function and u is a t-conorm function.

By this definition, it is reflected that:

1. The membership grade $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(T)$ is most "influenced" by the worst (the lowest) of the membership grades $\mu_{\tilde{U}}(T)$ and $\mu_{\tilde{V}}(T)$.
2. The membership grade $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(F)$ is most "influenced" by the worst (the greatest) of the membership grades $\mu_{\tilde{U}}(F)$ and $\mu_{\tilde{V}}(F)$.
3. The membership grade $\mu_{\tilde{U}\tilde{\wedge}_i\tilde{V}}(\perp)$ is most "influenced" by the worst (the greatest) of respectively the best (the lowest) of the membership grades $\mu_{\tilde{U}}(T)$ and $\mu_{\tilde{V}}(\perp)$, the best (the lowest) of the membership grades $\mu_{\tilde{U}}(\perp)$ and $\mu_{\tilde{V}}(T)$ the best (the lowest) of the membership grades $\mu_{\tilde{U}}(\perp)$ and $\mu_{\tilde{V}}(\perp)$.

Proposition 1. *The operators* $\tilde{\wedge}_i$ *are "almost compositional", i.e. with* $p, q \in P$, $\tilde{t}^*(p \wedge q) = \tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q)$ *always hold, except when* $q = \neg p$.

Proof. This proposition is proved by an exhaustive case study of the 'classical' conjunction as defined in Kleene's logic (operator $\wedge$).

i. If $\tilde{t}^*(p) = \{(T,1)\}$ and $\tilde{t}^*(q) = \{(T,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(T,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(1,1)), (F, u(0,0), (\perp, u(i(1,0), u(i(0,1), i(0,0)))))\} \\ &= \{(T,1)\} \text{ (Axioms i1, i2, i3, u1 and u4.)}\end{aligned}$$

ii. If $\tilde{t}^*(p) = \{(T,1)\}$ and $\tilde{t}^*(q) = \{(F,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(F,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(1,0)), (F, u(0,1)), (\perp, u(i(1,0), u(i(0,0), i(0,0))))\} \\ &= \{(F,1)\} \text{ (Axioms i1, i2, i3, u1, u3 and u4.)}\end{aligned}$$

iii. If $\tilde{t}^*(p) = \{(T,1)\}$ and $\tilde{t}^*(q) = \{(\perp,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(\perp,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(1,0)), (F, u(0,0)), (\perp, u(i(1,1), u(i(0,0), i(0,1))))\} \\ &= \{(\perp,1)\} \text{ (Axioms i1, i2, i3, u1, u3 and u4.)}\end{aligned}$$

iv. If $\tilde{t}^*(p) = \{(F,1)\}$ and $\tilde{t}^*(q) = \{(T,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(F,1)\}$
The proof of this case is analogous to the proof of case ii.

v. If $\tilde{t}^*(p) = \{(F,1)\}$ and $\tilde{t}^*(q) = \{(F,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(F,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(0,0)), (F, u(1,1)), (\perp, u(i(0,0), u(i(0,0), i(0,0))))\} \\ &= \{(F,1)\} \text{ (Axioms i2, u1, u2 and u4.)}\end{aligned}$$

vi. If $\tilde{t}^*(p) = \{(F,1)\}$ and $\tilde{t}^*(q) = \{(\perp,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(F,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(0,0)), (F, u(1,0)), (\perp, u(i(0,1), u(i(0,0), i(0,1))))\} \\ &= \{(F,1)\} \text{ (Axioms i1, i2, i3, u1, u2 and u4.)}\end{aligned}$$

vii. If $\tilde{t}^*(p) = \{(\perp,1)\}$ and $\tilde{t}^*(q) = \{(T,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(\perp,1)\}$
The proof of this case is analogous to the proof of case iii.

viii. If $\tilde{t}^*(p) = \{(\perp,1)\}$ and $\tilde{t}^*(q) = \{(F,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(F,1)\}$
The proof of this case is analogous to the proof of case vi.

v. If $\tilde{t}^*(p) = \{(\perp,1)\}$ and $\tilde{t}^*(q) = \{(\perp,1)\}$, then $\tilde{t}^*(p \wedge q) = \{(\perp,1)\}$

$$\begin{aligned}\tilde{t}^*(p) \tilde{\wedge}_i \tilde{t}^*(q) &= \{(T, i(0,0)), (F, u(0,0)), (\perp, u(i(0,1), u(i(1,0), i(1,1))))\} \\ &= \{(\perp,1)\} \text{ (Axioms i1, i2, i3, u1, u2, u3 and u4.)}\end{aligned}$$

□

As concrete definitions of conjunctive aggregation operators which are based on a t-norm function and on a t-conorm function (cf. Definition 4), the definitions of a 'logical' approach and an 'algebraic' approach are presented.

Definition 5. *The 'logical' conjunctive aggregation operator*

$$\tilde{\wedge}_l : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \rightarrow \tilde{\wp}(I^*) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \mathbin{\tilde{\wedge}_l} \tilde{V}$$

is defined by:

- $\mu_{\tilde{U}\tilde{\wedge}_l\tilde{V}}(T) = \min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(T))$
- $\mu_{\tilde{U}\tilde{\wedge}_l\tilde{V}}(F) = \max(\mu_{\tilde{U}}(F), \mu_{\tilde{V}}(F))$
- $\mu_{\tilde{U}\tilde{\wedge}_l\tilde{V}}(\bot) = \max(\min(\mu_{\tilde{U}}(T), \mu_{\tilde{V}}(\bot)), \min(\mu_{\tilde{U}}(\bot), \mu_{\tilde{V}}(T)),$
$\min(\mu_{\tilde{U}}(\bot), \mu_{\tilde{V}}(\bot)))$

The operator $\tilde{\wedge}_l$ is almost compositional by Proposition 1.

Definition 6. *The 'algebraic' conjunctive aggregation operator*

$$\tilde{\wedge}_a : \tilde{\wp}(I^*) \times \tilde{\wp}(I^*) \rightarrow \tilde{\wp}(I^*) : (\tilde{U}, \tilde{V}) \mapsto \tilde{U} \mathbin{\tilde{\wedge}_a} \tilde{V}$$

is defined by:

- $\mu_{\tilde{U}\tilde{\wedge}_a\tilde{V}}(T) = \mu_{\tilde{U}}(T)\mu_{\tilde{V}}(T)$
- $\mu_{\tilde{U}\tilde{\wedge}_a\tilde{V}}(F) = \mu_{\tilde{U}}(F) + \mu_{\tilde{V}}(F) - \mu_{\tilde{U}}(F)\mu_{\tilde{V}}(F)$
- $\mu_{\tilde{U}\tilde{\wedge}_a\tilde{V}}(\bot) = (\mu_{\tilde{U}}(T)\mu_{\tilde{V}}(\bot) + \mu_{\tilde{U}}(\bot)\mu_{\tilde{V}}(T) - \mu_{\tilde{U}}(T)\mu_{\tilde{V}}(\bot)\mu_{\tilde{U}}(\bot)\mu_{\tilde{V}}(T))$
$(1 - \mu_{\tilde{U}}(\bot)\mu_{\tilde{V}}(\bot)) + \mu_{\tilde{U}}(\bot)\mu_{\tilde{V}}(\bot)$

The operator $\tilde{\wedge}_a$ is almost compositional by Proposition 1.

4 Comparison and Applicability in Flexible Database Querying

In order to compare the presented definitions on the basis of their applicability in flexible database querying, a simple abstract database DB is considered, which consists of a collection of eight instances v_i, $i = 1, 2, \ldots, 8$. With respect to DB, a flexible query Q with two querying conditions C_1 and C_2 is considered. The extended possibilistic truth value which expresses the degree to which instance v_i satisfies constraint C_j is denoted as $\tilde{t}^*_{C_j}(v_i)$.

In Table 1 an example of the satisfaction degrees of the database instances for both individual constraints C_1 and C_2 is given.

Hereby, it is assumed that all instances satisfy constraint C_1 to the same (rather bad) extent ($\{(T, 0.4), (F, 0.6)\}$) and that for the instances $v_1, v_2, \ldots, v_5$ constraint C_2 is satisfied in a decreasing order, ranging from true ($\{(T, 1)\}$) to false ($\{(F, 1)\}$). Consequently, one may intuitively expect that the ranking of the instances $v_1, v_2, \ldots, v_5$ being considered as results of the query, is the following:

$$v_1 > v_2 > v_3 > v_4 > v_5$$

where v_1 is the best result.

Table 1. Example of $\tilde{t}_{C_j}(v_i)$.

v_i	$\tilde{t}^*_{C_1}(v_i)$	$\tilde{t}^*_{C_2}(v_i)$	$r(\tilde{t}^*_{C_1}(v_i))$	$r(\tilde{t}^*_{C_2}(v_i))$	average
v_1	$\{(T,.4),(F,.6)\}$	$\{(T,1)\}$	0.369	1	0.685
v_2	$\{(T,.4),(F,.6)\}$	$\{(T,.7),(F,.3)\}$	0.369	0.676	0.523
v_3	$\{(T,.4),(F,.6)\}$	$\{(T,.5),(F,.5)\}$	0.369	0.469	0.419
v_4	$\{(T,.4),(F,.6)\}$	$\{(T,.3),(F,.7)\}$	0.369	0.272	0.320
v_5	$\{(T,.4),(F,.6)\}$	$\{(F,1)\}$	0.369	0	0.185
v_6	$\{(T,.4),(F,.6)\}$	$\{(\bot,1)\}$	0.369	0.334	0.351
v_7	$\{(T,.4),(F,.6)\}$	$\{(F,.5),(\bot,1)\}$	0.369	0.229	0.299
v_8	$\{(T,.4),(F,.6)\}$	$\{(F,1),(\bot,1)\}$	0.369	0.167	0.268

Furthermore, it is assumed that for the instances v_6, v_7 and v_8 the degree of satisfaction is (possibly) undefined. For v_7 and v_8 it is additionally possible that the constraint is not satisfied at all ($\{(F,0.5),(\bot,1)\}$, resp. $\{(F,1),(\bot,1)\}$). Therefore, one may intuitively expect that the instances v_6, v_7 and v_8 are ranked as:

$$v_6 > v_7 > v_8$$

These intuitive rankings can mathematically be verified by considering the arithmetic average of the ranking values $r(\tilde{t}^*_{C_1}(v_i))$ and $r(\tilde{t}^*_{C_2}(v_i))$ of the possibilistic truth values for both constraints C_1 and C_2 (as illustrated in Table 1). Using the arithmetic average yields the global ranking:

$$v_1 > v_2 > v_3 > v_6 > v_4 > v_7 > v_8 > v_5$$

The results of the aggregations of the possibilistic truth values of Table 1 with the $\tilde{\wedge}$, $\tilde{\wedge}_l$ and $\tilde{\wedge}_a$ operators are presented in Table 2. In this table, the following shorthand notations are used: $r(v_i)$ denotes $r(\tilde{t}^*_{C_1}(v_i)\tilde{\wedge}\tilde{t}^*_{C_2}(v_i))$, $r_l(v_i)$ denotes $r(\tilde{t}^*_{C_1}(v_i)\tilde{\wedge}_l\tilde{t}^*_{C_2}(v_i))$ and $r_a(v_i)$ denotes $r(\tilde{t}^*_{C_1}(v_i)\tilde{\wedge}_a\tilde{t}^*_{C_2}(v_i))$.

Table 2. Application of the $\tilde{\wedge}$, $\tilde{\wedge}_l$ and $\tilde{\wedge}_a$ operators.

v_i	$\tilde{t}_{C_1}(v_i)\tilde{\wedge}\tilde{t}_{C_2}(v_i)$	$r(v_i)$	$\tilde{t}_{C_1}(v_i)\tilde{\wedge}_l\tilde{t}_{C_2}(v_i)$	$r_l(v_i)$	$\tilde{t}_{C_1}(v_i)\tilde{\wedge}_a\tilde{t}_{C_2}(v_i)$	$r_a(v_i)$
v_1	$\{(T,.4),(F,.6)\}$	0.369	$\{(T,.4),(F,.6)\}$	0.369	$\{(T,.4),(F,.6)\}$	0.369
v_2	$\{(T,.4),(F,.6)\}$	0.369	$\{(T,.4),(F,.6)\}$	0.369	$\{(T,.28),(F,.72)\}$	0.253
v_3	$\{(T,.4),(F,.5)\}$	0.411	$\{(T,.4),(F,.6)\}$	0.369	$\{(T,.2),(F,.8)\}$	0.178
v_4	$\{(T,.3),(F,.6)\}$	0.313	$\{(T,.3),(F,.7)\}$	0.272	$\{(T,.12),(F,.88)\}$	0.105
v_5	$\{(F,.6)\}$	0.196	$\{(F,1)\}$	0	$\{(F,1)\}$	0
v_6	$\{(F,.6),(\bot,.4)\}$	0.153	$\{(F,.6),(\bot,.4)\}$	0.153	$\{(F,.6),(\bot,.4)\}$	0.153
v_7	$\{(F,.6),(\bot,.4)\}$	0.153	$\{(F,.6),(\bot,.4)\}$	0.153	$\{(F,.8),(\bot,.4)\}$	0.094
v_8	$\{(F,.6),(\bot,.4)\}$	0.153	$\{(F,1),(\bot,.4)\}$	0.068	$\{(F,1),(\bot,.4)\}$	0.068

From the ranking values $r(v_i)$, $i = 1, 2, \ldots, 8$, it follows that by applying the conjunctive aggregation operator $\tilde{\wedge}$, the global ranking of the results is:

$$v_3 > v_1 = v_2 > v_4 > v_5 > v_6 = v_7 = v_8$$

This ranking is in contradiction with the expected 'intuitive' rankings $v_1 > v_2 > v_3 > v_4 > v_5$ and $v_6 > v_7 > v_8$: instance v_3 is considered to be the best solution for the query. Moreover, the ranking is too crude since no distinction is made between the (ranking of the) instances v_1 and v_2 and between the (ranking of the) instances v_6, v_7 and v_8.

From the ranking values $r_l(v_i)$, $i = 1, 2, \ldots, 8$, it follows that by applying the 'logical' conjunctive aggregation operator $\tilde{\wedge}_l$, the global ranking of the results is:

$$v_1 = v_2 = v_3 > v_4 > v_6 = v_7 > v_8 > v_5$$

This ranking is not in contradiction anymore with the expected 'intuitive' rankings $v_1 > v_2 > v_3 > v_4 > v_5$ and $v_6 > v_7 > v_8$, but is still in contradiction with the expected global ranking since $v_4 > v_6$. The ranking is also too crude since, for example, the instances v_1, v_2 and v_3 are considered to be of equal importance.

By observing the ranking values $r_a(v_i)$, $i = 1, 2, \ldots, 8$, it can be seen that the application of the 'algebraic' conjunctive aggregation operator $\tilde{\wedge}_a$ yields exactly the expected global ranking, i.e.

$$v_1 > v_2 > v_3 > v_6 > v_4 > v_7 > v_8 > v_5$$

Although the 'algebraic' conjunctive aggregation operator provides the best ranking of the presented operators, it is still not perfect. Consider for example two instances v_9 and v_{10}. Hereby, $\tilde{t}_{C_1}(v_9) = \tilde{t}_{C_1}(v_{10}) = \{(T, 0.4), (F, 1)\}$, whereas $\tilde{t}_{C_2}(v_9) = \{(T, 1)\}$ and $\tilde{t}_{C_2}(v_{10}) = \{(T, 1), (F, 0.3)\}$. Hence, the intuitive ranking of these instances is $v_9 > v_{10}$ (this is also reflected by the average range which is respectively 0.630 and 0.556). The application of the $\tilde{\wedge}_a$ operator however, yields

$$\tilde{t}_{C_1}(v_9)\tilde{\wedge}_a\tilde{t}_{C_2}(v_9) = \tilde{t}_{C_1}(v_{10})\tilde{\wedge}_a\tilde{t}_{C_2}(v_{10}) = \{(T, 0.4), (F, 1)\}$$

which implies the ranking $v_9 = v_{10}$, which is still too crude.

In general, two instances are indistinguishable with respect to the 'algebraic' operator $\tilde{\wedge}_a$ if at least one of the arguments a and b of the 'algebraic' t-conorm $u(a, b) = a + b - ab$ equals 1 (because 1 is the neutral element for multiplication). Such instances are also indistinguishable with respect to the 'logical' operator $\tilde{\wedge}_l$ (because $\forall\ a \in [0, 1] : u(a, 1) = \max(a, 1) = 1$). Moreover, as shown by the example, not all indistinguishable instances for the $\tilde{\wedge}_l$ operator are indistinguishable for the $\tilde{\wedge}_a$ operator. This leads to the conclusion that, with respect to the ranking function r, the $\tilde{\wedge}_a$ operator is best suited for conjunctive aggregation of extended possibilistic truth values in flexible database querying.

5 Conclusions

Extended possibilistic truth values provide a means to handle the (un)certainty about the truth of a proposition and can be used to model constraint satisfaction in flexible database querying. This paper deals with the conjunctive aggregation of extended possibilistic truth values. Three definitions for a conjunctive aggregation operator are presented and compared with each other, using the ranking function r. Each definition has different capabilities to rank the possible alternatives in the results of a flexible database query. The first definition is obtained as an extension of Kleene's logical conjunction operator by applying Zadeh's extension principle. Its ranking capabilities are too crude and very limited because a resulting ranking can be in contradiction with the expected 'intuitive' ranking. The second and third definition are respectively based on the logical t-(co)norm operation and on the algebraic t-(co)norm operation. These definitions do not result in a non-intuitive ranking anymore, but are still too crude. From the three definitions, when applying the ranking function r, the definition based on the algebraic t-(co)norm is less crude than the other ones, has the best ranking capabilities and is therefore the best suited for conjunctive aggregation of extended possibilistic truth values in flexible querying and answering systems.

References

1. De Tré, G., De Caluwe, R., Van der Cruyssen, B.: A Generalised Object-Oriented Database Model. In: Recent Issues on Fuzzy Databases, Bordogna, G., Pasi, G. (eds.) Physica-Verlag, Heidelberg, Germany (2000) 155–182
2. Kuper, G.M., Libkin, L., Paredaens, J. (eds.): Constraint Databases. Springer-Verlag, Heidelberg, Germany (2000)
3. Dubois, D., Prade, H.: Certainty and Uncertainty of (Vague) Knowledge and Generalised Dependencies in Fuzzy Databases. In: Proc. of International Fuzzy Engineering Symposium '91, Yokahoma, Japan (1991) 239–249
4. Baldwin, J.F., Zhou, S.Q.: A Fuzzy Relational Inference Language. Fuzzy Sets and Systems **14** (1984) 155–174
5. Van Schooten, A.: Ontwerp en implementatie van een model voor de representatie en manipulatie van onzekerheid en imprecisie in databanken en expert systemen. Ph.D. Thesis (in Dutch), Ghent University (1988) (unpublished)
6. Van Gyseghem, N.: Imprecision and uncertainty in the UFO Database Model. Journal of the American Society for Information Science **49** 3 (1998) 236–252
7. de Tré, G., de Caluwe, R.: Application of Extended Possibilistic Truth Values in Multimedia Database Systems. In: Proc. of ECSQARU-2001 workshop on Management of uncertainty and imprecision in multimedia information systems, Toulouse, France (september 2001)
8. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems **1** 1 (1978) 3–28
9. Dubois, D., Prade, H.: Possibility Theory. Plenum Press, New York, USA (1988)
10. Prade, H., Testemale, C.: Generalizing Database Relational Algebra for the Treatment of Incomplete or Uncertain Information and Vague Queries. Information Sciences **34** (1984) 115–143

11. Bordogna, G., Pasi,G., Lucarella, D.: A Fuzzy Object-Oriented Data Model for Managing Vague and Uncertain Information. International Journal of Intelligent Systems **14** 7 (1999) 623–651
12. Prade, H.: Possibility sets, fuzzy sets and their relation to Lukasiewicz logic. In: Proc. of the 12th International Symposium on Multiple-Valued Logic, Paris, France (1982) 223–227
13. de Tré, G.: Extended Possibilistic Truth Values. International Journal of Intelligent Systems **17** (2002) 427–446
14. Rescher, N.: Many-valued logic. McGraw-Hill, New York, USA (1969)
15. Zadeh, L.A.: The concept of a linguistic variable and its application to approximate reasoning I. Information Sciences **8** (1975) 199–251
16. Klir, G.J., Yuan, B.: Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall Inc., New Jersey, USA (1995)
17. Dubois, D., Prade, H.: Possibility theory, probability theory and multiple-valued logics: A clarification. Annals of Mathematics and Artificial Intelligence **32** (2001) 35–66

Interactive Query Formulation in Semistructured Databases

Agathoniki Trigoni

Athens University of Economics and Business
Department of Informatics
ntrig@nbg.gr

Abstract. The use of large amounts of distributed and heterogeneous information has become extremely cumbersome; this difficulty is mainly related to exploring the data, rather than actually storing or exchanging it. The user who is interested in small bits of information is getting more and more confused when having to dig under a large volume of diverse and more importantly semi-structured data. In this paper, we propose an interactive and adaptive framework that guides the user in the search for data, by disclosing only a part of the underlying information at a time. It first provides the user with a high-level view of the raw data and gradually adapts to his/her needs in order to offer a refined answer. The proposed model offers the possibility to query a semistructured database based on general schema-related constraints imposed by the user or identified by the system, but without specific knowledge of the underlying metadata. This is achieved by receiving initially an *amorphic* query, which may consist of one or more basic paths, and helping the user to refine it gradually to a specific semistructured query, expressed in a language like XQuery or Lorel.

1 Introduction

The queries processed in the areas of databases and information retrieval are tailored for the needs of the underlying systems. Regarding databases, query constructs vary depending on whether the context is an object-oriented [1], relational [2], deductive [3] or semistructured database [4,5,6]. In general, prior knowledge of the database structure is required, even if it is limited or is inferred automatically, as in the case of semistructured databases [7,8,9]. On the other hand, searches in the context of information retrieval have a different style; they are usually based on identifying a set of keywords in a series of documents, and using a variety of criteria and ranking algorithms to sort the resulting documents based on their relevance to the keywords [10].

Neither of the two query styles provides the most natural way of searching. The query framework proposed in this paper takes into account the following points:

- Users can usually contribute to the query input in a more intelligent way than by providing a few keywords.
- A system is not user-friendly when it requires a detailed knowledge of the database structure, especially in the presence of large amounts of heterogeneous data.

- The user would not like to miss information because of his lack of knowledge about the schema. Query languages in semistructured systems currently allow the user to avoid type errors when not complying with the explicit or implicit schema. However, the user ends up receiving only a part of the information requested.
- Users usually have a high-level knowledge of the database structure, based on their natural perception of data correlations.
- The lack of knowledge about the database structure does not preclude the need to enforce specific selection criteria on the results.

Structured queries require knowledge of schema information, or else fail to deliver complete and accurate results. IR-style searches do not exploit but a small part of the user's knowledge about the data (mainly keywords), and result in sets of documents that cannot be filtered using detailed criteria. The idea behind this paper is that the tradeoff between allowing *natural* queries and receiving quality (complete and accurate) results, could be compromised if we adopted an interactive and adaptive query model.

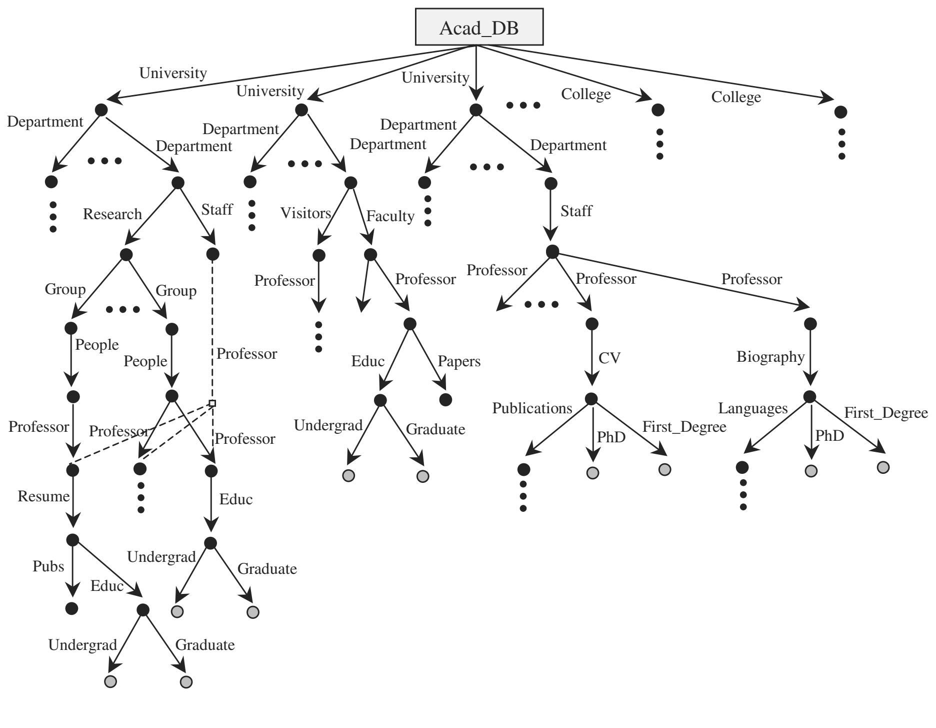


Fig. 1. XML data: Academic information about universities and colleges

Consider for instance an XML tree that represents academic information about universities and colleges (figure 1). The root of the tree named `Acad_DB` has several `University` labels and each one of them is organised in a different way. Say that a Scholarships Foundation is interested in funding the Computer Science department of a university with strong background in mathematics. Hence, they look for the department with the maximum number of academics having studied maths in their first degree and computer science in their graduate studies. That is, they search for quite specific information without having specific knowledge of the data organisation, but merely a high level view of how universities are organised in general.

Using an IR approach and providing keywords like *maths*, *professor*, *informatics* they might be able to receive relevant Web documents, but still it would be difficult to identify automatically the department that satisfies their selection criterion. Since there is a need for an exhaustive search in all universities, well-known semistructured queries cannot be trusted, since they might exclude interesting information from the result, without even sending an error message, e.g. type conflicts. Our objective is the tight coupling of knowledge discovery and query processing, as a means of providing the user with the best answer possible, exploiting existing high-level knowledge of the domain context but also enabling low-level filtering techniques.

In what follows, we present the functionality of the proposed model. In section 2 we discuss the kinds of schema and semantic information used by the interactive query processor. This information is either discovered in advance or mined upon the request of a user. The algorithm that guides the interactive process is proposed in section 3. A concrete example that demonstrates the process of gradual query refinement is presented in section 4. Based on the example, we discuss the advantages of our framework compared to more traditional query or IR engines, and show how it could potentially be extended in order to take into account new kinds of knowledge about the data (section 5). Finally, section 6 refers to previous work related to our framework and section 7 presents concluding remarks highlighting the novel points of the proposed model.

2 Data Organization: Schema, Morphology, and Semantics

In most existing database systems, queries, which are expressed in declarative languages, provide information about the location of the data requested, and include operations like filtering, grouping and ordering. In our model, it is not necessary to specify explicitly the location of data; instead a number of new search criteria can be used that are metadata-related and allow for the expression of a wider variety of queries. Prior knowledge of metadata is not required, but metadata information can be used as a requirement (or filter) for the expected results. Metadata information is also used by the query processor, e.g. in order to justify that the user requirements are not satisfiable, as it will be shown in section 4.

The kinds of metadata information used either as search criteria (by the user), or as guides for query refinement (by the query processor), are presented below. Similar information has also been used for the semantic optimization of semistructured queries [11,12]. In what follows, paths are denoted as $\mathtt{p}$, $\mathtt{p_i}$ or $\mathtt{p'_i}$, and represent either specific paths (e.g. `Student.course.name`) or path patterns (e.g. `Student.course.`$*$, `Student.`$*$`(2,4).name`). The symbol $*$ represents a sequence of zero or more labels, whilst $*(\mathtt{m},\mathtt{n})$ represents a sequence of `m` to `n` labels. The notation $\mathtt{lab_1}|\mathtt{lab_2}$ denotes that either of the two labels can be used; for example `Person.name|job` includes paths `Person.name` and `Person.job`. More formally, a path (or path pattern) is a sequence of zero or more elements of the form:

$$\begin{aligned}\texttt{element} ::= \; & \texttt{label} \\ | \; & \texttt{label}_1 | \dots | \texttt{label}_k \\ | \; & * \\ | \; & *(\texttt{m}, \texttt{n})\end{aligned}$$

where $\texttt{label}, \texttt{label}_1, \dots, \texttt{label}_k$ are names of labels in a semistructured database.

The expression $p \subseteq p'$ expresses that path p satisfies path p', i.e. all specific paths (sequences of labels) that are instances of p are also instances of p'.

1. *Sequence patterns*. Consider for instance a rule of the form

$$\forall p_1, p_2.\ p_1 \subseteq p'_1 \wedge p_1.p_2 \Rightarrow p_2 \subseteq p'_2$$

   This rule expresses that if a path p_1 satisfies the pattern p'_1 and p_1 is followed by p_2, then the latter satisfies the pattern p'_2.
   This metadata information could be used by the user to search for all paths p_1 that satisfy the rule above. For instance, a user might be interested in all *authors* that have written only *fiction* and hence there are no other labels (e.g. *novel*, *essay*) that start from these author nodes.
   The rule above might also be useful to the query processor to inform the user about the sequence patterns of a path p''_1, such that $p''_1 \subseteq p'_1$ or to justify why a query $p''_1.p''_2$, such that $p''_1 \subseteq p'_1$ and $\texttt{not}(p''_2 \subseteq p'_2)$ would yield no results.
2. *Sibling patterns*. Paths in semistructured databases often exhibit properties that are related to the properties of sibling paths, i.e. of paths that start from a common parent node. Consider for instance a rule of the form

$$\forall p_1, p_2.\ p_1 \subseteq p'_1 \wedge p_2 \subseteq p'_2 \wedge p_1.p_2 \Rightarrow \exists p_3.\ p_3 \subseteq p'_3 \wedge p_1.p_3$$

   That is, if path p_1 is followed by p_2 then it is also followed by at least one path p_3, where $p_1 \subseteq p'_1$, $p_2 \subseteq p'_2$ and $p_3 \subseteq p'_3$.
   The query processor could use this semantic information in order to present the properties of a given path p_1 or $p_1.p_2$ and thus help the user to refine it to a specific query.
   The rule itself could be presented by the user as a query, e.g. find all paths p_1 that satisfy (or not) the rule. This query could probably be expressed using existing query languages for semistructured databases, but if the consequent of the rule was a universal quantification, this would be impossible.
3. *Shortcut properties* Consider for instance the rule

$$\begin{aligned}&\forall p_1, p_2.\ p_1 \subseteq p'_1 \wedge p_2 \subseteq p'_2 \wedge p_1.p_2 \Rightarrow \\ &\exists p_3.\ p_3 \subseteq p'_3 \wedge \texttt{end\_node}(p_1.p_2) = \texttt{end\_node}(\texttt{p1.p3})\end{aligned}$$

   The rule expresses that a path $p_1.p_2$ leads to the same point as at least one alternative path $p_1.p_3$. If p_3 is shorter (has fewer labels) than p_2 then the former is a shortcut of the latter.
   The query processor might propose the use of a shorter path than the one specified initially, in order to reveal potentially interesting schema information and thus guide the user to express a more appropriate query. Shorter paths might also contribute to the more efficient execution of the initial query.

On the other hand, the user might be interested in paths that satisfy the pattern $\mathtt{p}'_1.\mathtt{p}'_2$ and that have at least one alternative path. For instance, it might be interesting to search for all countries such that if they export products to a certain country, they also import other products from it.

4. *Specificity* Consider for instance the following rule:

$$\forall \mathtt{p}.\ \mathtt{p} \subseteq \mathtt{p}' \Rightarrow \mathtt{p} \subseteq \mathtt{p}''$$

It expresses the metadata information that paths that satisfy the pattern $\mathtt{p}'$ also satisfy the more specific pattern $\mathtt{p}''$. If such a rule is true in the database, then the query of a user that involves a path $\mathtt{p}_1 \subseteq \mathtt{p}'$ should become more specific, say $\mathtt{p}_2$ so that $\mathtt{p}_2 \subseteq \mathtt{p}''$. Exploiting metadata information, the query processor helps the user to refine his/her queries. Notice that this information is only exploited by the query processor and is not usually expressed as a query by the user.

5. *Topological properties* Topological information refers to the relevant location of a path, e.g. to its distance from a certain node. For instance the semantic information $\mathtt{distance(anchor\_node, p)\ in\ R}$ expresses the fact that the number of labels leading from $\mathtt{anchor\_node}$ to the beginning of path $\mathtt{p}$ are in range $\mathtt{R}$, e.g. R=[2,4]. The information $\mathtt{bottom\_distance(p)\ in\ R}$ denotes that the distance of the end of path $\mathtt{p}$ to a leaf (to which it leads) is in range $\mathtt{R}$.
A user might be interested in paths that are close to leaf nodes, since leaves contain values upon which one may apply selection criteria. For instance, a user might query all paths $\mathtt{p}$ such that $\mathtt{bottom\_distance(p)\ in\ [1,3]}$.
If a user has very little knowledge of the structure of an XML tree, he might be interested in discovering gradually labels that are in the initial levels, and hence denote general ideas, or in labels close to leaves, which express what kind of values are stored in the database.

6. *Cardinality and Length of patterns*
Users are often interested in paths that occur a certain number of times in the database, independent of their location. The expression $\mathtt{number(p)\ in\ R}$ denotes that the cardinality of paths of the form $\mathtt{p}$ are in range $\mathtt{R}$. For instance, in a survey about labour conditions in U.K. it is expected that labels that link individuals with jobs should be almost as many as the people of the sample. Hence, even if we have no prior knowledge of the specific label, its cardinality can be used as a starting point to enter and further explore the database. The problem of finding frequent subsequence patterns of the form $*.\mathtt{X}_1.*.\mathtt{X}_2.*...*.\mathtt{Xn}.*$, where $\mathtt{X_i}$ is a subsequence of at least $\ell_\mathtt{i}$ labels ($\mathtt{i} = 1, \ldots, \mathtt{n}$), has been studied in the context of combinatorial pattern discovery for scientific data [13]. In that context, the notion of maximum edit distance (as a dissimilarity measure) is used so that the support of a pattern is also increased by sequences that approximately satisfy a pattern (within edit distance $\mathtt{d}$). This dissimilarity measure can also be used in our work, in order to allow a user to express queries of the form: "Find all subpaths of length 3 (i.e. $\mathtt{lab_1.lab_2.lab_3}$) that occur at least $\mathtt{100}$ times in the database within distance 1".
The length of paths satisfying a pattern can be used in order to avoid cyclic paths, or indirect paths irrelevant to the user requirements. For instance, in a genealogic XML graph, it is sufficient to use paths of length at most 3 in order to find the close relatives of a certain person. If a user specifies a query pattern satisfiable by many paths of completely different lengths, the query processor should notify the user of this information, in order to help him/her refine the query.

7. *Pattern Similarity* In our discussion about pattern cardinality (or support), we mentioned the problem of searching for frequent subsequences of labels of length ℓ within edit distance `d`. The user searches for frequent patterns of a minimum length, allowing for a certain degree of sequence dissimilarity. The dissimilarity measure can also be used in a similar problem, in which the user searches for all paths that contain a subsequence similar to an input path pattern. This need may occur for two reasons: i) a user might be unable to find any paths in the database satisfying the original pattern and ii) even if a pattern is found in the database, the user would like to cover the possibility of ignoring very similar and equally interesting patterns. Hence, given an initial path pattern, our framework provides the opportunity of searching the semistructured data for all similar paths within distance `d`, i.e. that satisfy the pattern if we modify (insert, delete or substitute) at most `d` of their labels.

It is interesting to notice that the kinds of metadata information defined above could be combined in several ways. For instance, a user might be interested in paths from a source to a target node, such that the number of alternative paths of length ℓ that link the two nodes are more than `n`. Likewise, the system could combine this knowledge in order to justify why a certain query is expected to yield no results, or as a way to aid in the refinement of the query. In the following section, we show the interactive process that exploits the metadata information defined above, namely path sequences, inheritance properties, connectedness and shortcuts, specificity rules, topological requirements, cardinality and length properties, as well as pattern similarity.

3 Interactive Search Algorithm

The idea behind the interactive search algorithm is that the user cannot express initially a specific query, but gradually refines a general one based on either metadata requirements that s/he enforces or metadata information about the schema that the query processor reveals.

The detailed algorithm is given below:
Step 1: The user initially presents a query, as a set of path templates (patterns).
Step 2: Repeat substeps 2.1 and 2.2

- *Substep 2.1:* The system uses metadata information about sequence and inheritance patterns, or specificity rules, in order to justify the absence of some input patterns, as well as to refine others. If metadata information relevant to the patterns is not available, the query processor navigates over the semistructured database in order to identify new knowledge and store it in the warehouse.
- *Substep 2.2:* The user adds, modifies, deletes, or accepts the resulting path templates.

Until the resulting patterns from substep 2.1 are not altered at all by the user in substep 2.2.
Step 3: Given the refined patterns, the user may proceed in either of the following ways:

1. *By looking for metadata information relevant to one or more of the patterns.*
   The user may select one or more of the following metadata information:
   - Sequence patterns
   - Sibling patterns

  - Connectedness and shortcut properties
  - Topology information
  - Cardinality and Length information
  - Similar patterns within edit distance `d`

  Based on the resulting metadata information, the user may modify some of the existing paths, or add new ones. The algorithm is then continued from step 2.
2. *By enforcing metadata conditions as requirements in order to further refine one or more patterns.*
   The requirements might concern one or more of the following metadata conditions:
   - Sequence patterns
   - Sibling patterns
   - Connectedness and shortcut properties
   - Topology information
   - Cardinality and Length information

   The metadata conditions result in more refined patterns, that the user might then accept, modify, reject or add new ones. The algorithm is then continued from step 2.
3. *By asking the system to present the user with all specific paths in the database that actually satisfy one or more of the refined paths.* In fact, the user may only be interested in the enumeration of ℓ consecutive labels preceding or following a segment (subpath) of a given pattern.
   In this case the user may select specific paths, or add, change, or delete some of the initial path patterns. The algorithm is then continued from step 2.
4. *If the patterns are already specific, the interactive process ends and the user may express a detailed query in a language like Lorel or XQuery.*

The functionality of the algorithm presented above is demonstrated through the use of an example in section 4.

4 Searching and Querying a Semistructured Database: Example

Consider the database consisting of academic information about universities and colleges (figure 1), which was first mentioned in section 1. The user is interested in identifying the university department with the maximum number of academics having studied maths in their first degree and computer science in their graduate studies.
Step 1

- *User:* The user registers interest for paths of the form:

$$\mathtt{P_1: *.University.*.Professor.*.Educ}$$

Step 2

- *System:* The system applies the specificity rule that all paths should start with the only entry point to the database `Acad_DB` and converts $\mathtt{P_1}$ to $\mathtt{P_2}$.

$$\mathtt{P_2: Acad\_DB.University.*.Professor.*.Educ}$$

  It also uses the sequence pattern that all children of `University` labels are named `Department` to convert $\mathtt{P_2}$ to $\mathtt{P_3}$.

$$\mathtt{P_3: Acad\_DB.University.Department.*.Professor.*.Educ}$$

Step 3

- *User:* The user queries the length of subpaths `Department.` $*$ `.Professor` in $\mathtt{P}_3$ since he is interested in closely linked `Department` and `Professor` nodes. A long path between a `Department` and a `Professor` would signify that the latter does not belong to the former, but is rather remotely associated with it. If such a long subpath leads to the same `Professor` as a shorter path, then the user would be interested in considering only the shortcut.
- *System:* The answer is that we have paths of length 3 and 5.
- *User:* The user wonders why the subpaths `Department.` $*$ `.Professor` are of two different lengths, and attempts to investigate the properties of the longer ones. If `Professor` nodes that are reached by paths of length 5 were also reached by paths of length 3, then the former paths could be ignored. Thus the user puts the condition that the length of `Department.` $*$ `.Professor` in $\mathtt{P}_3$ should be 5 and queries any connectedness properties of the resulting paths.
- *System:* The resulting subpaths of length 5 have the form $\mathtt{P}_4$:

  $$\mathtt{P_4 : Department.Research.Group.People.Professor}$$

  Regarding connectedness properties of $\mathtt{P}_4$, all nodes reached by

  $$\mathtt{Department.Research.Group.People.Professor}$$

  are also reached by

  $$\mathtt{Department.Staff.Professor}$$

Step 2

- *User:* Hence the user decides to refine $\mathtt{P}_3$ by specifying that only one label separates `Department` and `Professor`:

  $$\mathtt{P_5 : Acad\_DB.University.Department.} * (1,1)\mathtt{.Professor.} * \mathtt{.Educ}$$

- *System:* No refinement is possible.

Step 3

- *User:* An enumeration of subpaths $\mathtt{Department.} * (1,1)\mathtt{.Professor}$ (of path $\mathtt{P}_5$) is requested.
- *System:* The answer is:

  $$\mathtt{Department.Faculty|Staff|Visitors.Professor}$$

Step 2:

- *User:* The user is not interested in `Visitors`, so s/he refines pattern $\mathtt{P}_5$ to $\mathtt{P}_6$. The user also adds the new pattern $\mathtt{P}_7$ taking into account that education information about a professor might not necessarily follow an `Educ` node. Other relevant keywords could be `CV`, `Biography` or `Curriculum_Vitae`.

  $$\mathtt{P_6 : Acad\_DB.University.Department.Faculty|Staff.Professor.} * \mathtt{.Educ}$$
  $$\mathtt{P_7 : University.} * \mathtt{.CV|Biography|Curriculum\_Vitae}$$

– *System:* The system refines $\mathtt{P}_6$ and $\mathtt{P}_7$ to $\mathtt{P}_8$ and $\mathtt{P}_9$ respectively. Notice that label `Curriculum_Vitae` is not included in $\mathtt{P}_9$.

$\mathtt{P}_8$: `Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ`
$\mathtt{P}_9$: `Acad_DB.University.Department.*.Professor.CV|Biography`

Step 3

– *User:* The user searches for sibling patterns in $\mathtt{P}_8$ and $\mathtt{P}_9$. The reason is that s/he suspects that graduate and undergraduate studies (independent of their actual labels) are likely to occur together under either `Educ` or `CV|Biography`.
– *System:* A sibling pattern was identified in $\mathtt{P}_8$:

$$\forall \mathtt{p}.\mathtt{p} \subseteq \mathtt{Educ} \wedge \mathtt{p.Undergrad} \Rightarrow \mathtt{p.Graduate}$$

– *User:* Based on previous knowledge, the user refines path $\mathtt{P}_9$ by replacing its subpath `Department.*.Professor` with `Department.Faculty|Staff.Professor`. Having no further specific requirements, s/he then asks for an enumeration of all labels that follow `CV` or `Biography` in $\mathtt{P}_9$.
– *System:* `PhD|First_Degree|Publications|Languages`

Step 2 New paths presented by user.

$\mathtt{P}_{10}$: `Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ.Undergrad`
$\mathtt{P}_{11}$: `Acad_DB.University.Department.Faculty|Staff.Professor.*.Educ.Graduate`
$\mathtt{P}_{12}$: `Acad_DB.University.Department.Faculty|Staff.Professor.CV|Biography.First_Degree`
$\mathtt{P}_{13}$: `Acad_DB.University.Department.Faculty|Staff.Professor.CV|Biography.PhD`

Step 3: The user queries the distance of these paths from leaves and the system confirms that it is 0.

Hence the user may use these paths to form the following query:

```
Q1 = (select dept : x.dept, no_of_prof : count(x.prof)
   from
  (select dept : Acad_DB.University.Department, prof : p
  from Acad_DB.University.Department.Faculty|Staff.Professor as p
  where p.*.Educ.Undergrad contains Maths
  and p.*.Educ.Graduate contains Computer|Inform
  union
  select dept : Acad_DB.University.Department, prof : p
  from Acad_DB.University.Department.Faculty|Staff.Professor as p
  where p.CV|Biography.First_Degree.university contains Math
  and p.CV|Biography.PhD contains Computer|Inform)
     as x
   group by dept : x.dept)
```

$\mathtt{Q}_1$ returns all departments paired with the number of professors that have studied mathematics in their first degree and computer science or information techonology in their graduate studies. The following query selects the department with the maximum number of these professors.

```
select dept : y.dept
from Q1 as y
where y.no_of_prof =
  max(select z.no_of_prof from Q1 as z)
```

5 Advantages of the Interactive Search Algorithm

The example given above demonstrates a number of advantages of the proposed interactive search algorithm to traditional querying or IR techniques.

1. The user does not usually have a prior knowledge of schema information. Even if this information is revealed, it might be very confusing for the user to assimilate and handle. Our model provides the mechanism to gradually acquire relevant schema information to his/her information needs.
2. The user is enabled to do a search not only based on content criteria (keywords), neither only in exact location criteria (traditional queries). The model allows us to use more meaningful schema-related criteria, e.g. sequence, sibling, topological, connectedness, length and cardinality path constraints.
   - By looking for schema-related patterns, the user can gradually find his way to the paths that are relevant to his/her search. It is often more helpful to find our way through a graph, if we know general structure characteristics, rather than a detailed account of its paths.
   - By enforcing schema-related constraints, the user manages to limit the scope of the database, keeping a view only to the parts that satisfy the constraints. This is a powerful mechanism for users who do not know the exact location of the data, but have some intuitive knowledge of the constraints satisfied by the interesting paths. Since these constraints are structure- rather than value-related, they cannot easily be expressed in the *where* clause of a *select* query. However, they are very useful in refining the paths presented initially by the user.
   - As a last resort the user may request an enumeration of all possible paths satisfying a certain pattern. Such an enumeration is useful, because it is done in a local context (after paths are already refined and the scope of the database is limited). If it was performed in order to reveal the organisation of a whole semistructured database, the amount of information would be overwhelming and rather useless.
3. The iterative nature of the algorithm allows the user to readapt his search based on the additional knowledge acquired in previous steps. The user may interrupt the iterative process, only when s/he is satisfied with the expected degree of completeness of the result.
4. The interactive algorithm is extendible in two ways: First, new kinds of patterns or constraints could be searched or enforced respectively, as long as they are supported in programming terms. For instance, distributed, or mobile data sources could be queried based on semantic information about the distribution or movement patterns. Second, ontological information could enhance the functionality of the system significantly [14,15,16]. For instance, based on labels defined by the user, the system may reveal synonyms or relevant concepts, that could be parts of equally interesting paths. Ontologies have been traditionally used to enhance IR searches; however, their use should also be beneficial in the process of defining more conventional *select* queries.

5. Our model is a hybrid system that combines characteristics of both IR and conventional query models. The user may start with a few keywords (as in IR) and by gradually identifying or imposing structural database constraints, s/he may then refine the search gradually, until a detailed query is formed. Hence, the user may satisfy his need for applying detailed selection criteria, despite the initial lack of specific knowledge about the database schema. This is a very common need, and hence our model is expected to be useful for a variety of applications.

6 Related Work

The World Wide Web presents a great number of challenges from the viewpoint of database theory [17], including storing and querying data, imposing constraints and mining patterns [18]. As is stated in [17], 'a database is a polished artifact', compared to 'a free-evolving, everchanging' collection of data sources constituting the Web. Hence, there is a need to devise flexible models for finding and extracting knowledge either by querying raw XML-ized data or by mining patterns from semistructured databases or their corresponding schemata (DTDs).

Several languages have recently been proposed for querying XML databases [4,5, 19,6]. On the other hand, many research groups have focused on integrity or typing constraints applying to the semistructured model [20,21]. These constraints have either been used for query optimization or as a means of preventing anomalies during updates or data integration. However, very little work has focused on a flexible mechanism for querying information with the aid of constraints. Amongst the few tools available for helping the user query a semistructured database, it is worth mentioning Document Type Descriptors (DTDs). A DTD, which may optionally accompany an XML document, serves the role of a schema specifying the internal structure of the document. A significant amount of work has studied the inference of a DTD from one or more documents, as well as the problem of deciding whether a document conforms to a given DTD [7,8,9]. XTRACT, a recent system for inferring a DTD schema, applies the Minimum Description Length (MDL) principle in order to generate both concise and intuitive DTDs containing regular expressions. Semantically meaningful DTDs can be used as a starting point in order to identify constraints, e.g. sequence or sibling patterns, using a variety of data mining algorithms.

This paper proposes a framework for interactive query formulation using metadata-related conditions; however, it does not consider the problem of actually identifying constraints by defining new mining algorithms. The latter problem has been addressed in the literature under the form of mining sequential patterns [22,23,24] and association rules [25] or combinatorial pattern discovery [13]. If we consider paths of a semistructured database as sequences of labels, then we may apply well-known algorithms in order to find frequent subsequences of labels as well as (sequence or sibling) association rules. This is a vertical way of deriving sequences from a semistructured database, as opposed to the horizontal view of the database corresponding to DTDs. Mining association rules or sequential patterns for our purposes may either take place initially, or at the time of a request. In the latter case, our framework could benefit from algorithms in which users play an active role in guiding the mining process [26,27].

Our framework is equally relevant and could benefit from previous work on discovering frequent tree expressions in documents [28,29]. Typical structures in [28] can be used as a 'table-of-content' for gaining the general information of a source, or as a 'road map' for browsing and querying a source. The focus of our work is again not on mining such typical structures, but on demonstrating the use of several kinds of metadata information as guides in the querying process.

Finally, helping users find their way in a semistructured database has also been considered in [30,31] but following a different approach. In particular, they have not considered structural properties and constraints as a means of adding flexibility to the querying interface, but instead exploited path information generated and stored in dynamic schemata (DataGuides). DataGuides are similar to DTDs in that they provide concise and convenient summaries of semistructured databases. Unlike static schemata, DataGuides conform to the data, rather than forcing data to conform to DataGuides. Hence, they always represent the current state of the database and play a significant role especially in contexts where it is implausible to specify and maintain a schema. DataGuide summaries may be used in our framework as concise and rich sources for mining metadata constraints efficiently.

7 Conclusion

In this paper, we proposed an interactive and adaptive model for querying semistructured databases. Starting from an IR-style query, the user may discover interesting schema information, and gradually refine his/her search to a conventional *select* query. Our model has a dynamic notion of context, starting from high level information to raw data and limiting the scope from the whole database, to local subpaths or subtrees. Adhering to the human sense of searching things, it is not only based on value or location criteria, but it equally exploits a variety of high level schema-related knowledge. This information is stored in a warehouse, which is maintained based on an event-driven mechanism. The proposed model should be very useful for searching semistructured databases, when results need to be filtered based on detailed criteria and, hence, low-granularity results (e.g. documents) are not acceptable. In general, it should be useful for all applications with an underlying semistructured database, but its benefits would be particularly obvious in the context of searches over the Web.

References

1. Cattell et al, R.: The Object Data Standard: ODMG 3.0. Morgan Kaufmann (2000)
2. Chaudhuri, S., Shim, K.: Optimizing queries with aggregate views. In: Extending Database Technology. (1996) 167–182
3. Chakravarthy, U., Grant, J., Minker, J.: Foundations of semantic query optimization for deductive databases. In: Foundations of Deductive Databases and Logic Programming. (1988) 243–273
4. Abiteboul, S.: Querying semi-structured data. In: Intl Conf on Database Theory. (1997) 1–18
5. Buneman, P.: Semistructured data: a tutorial. In: Symposium on Principles of Database Systems. (1997)

6. Evangelista-Filha, I., Laender, A., Silva, A.: Querying semistructured data by example: The QSByE interface. In: Intl Workshop on Information Integration on the Web (WIIW). (2001) 156–163
7. Buneman, P., Davidson, S., Fernandez, M., Suciu, D.: Adding structure to unstructured data. In: Intl Conf on Database Theory. (1997) 336–350
8. Nestorov, S., Abiteboul, S., Motwani, R.: Inferring structure in semistructured data. In: Workshop on Management of Semistructured Data. (1997)
9. Nestorov, S., Abiteboul, S., Motwani, R.: Extracting schema from semistructured data. In: ACM Intl Conf on Management of Data. (1998) 295–306
10. Kobayashi, M., Takeda, K.: Information retrieval on the web. ACM Computing Surveys **32** (2000) 144–173
11. Buneman, P., Fan, W., Simeon, J., Weinstein, S.: Constraints for semistructured data and xml. SIGMOD Record **30** (2001)
12. Buneman, P., Fan, W., Weinstein, S.: Query optimization for semistructured data using path constraints in a deterministic data model. In: Workshop on Database Programming Languages. (1999) 208–223
13. Wang, J., Chirn, G.W., Marr, T., Shapiro, B., Shasha, D., Zhang, K.: Combinatorial pattern discovery for scientific data: Some preliminary results. In: ACM SIGMOD Intl Conf on Management of Data. (1994) 115–125
14. Chandrasekaran, B., Josephson, J., Benjamins, V.: What are ontologies, and why do we need them? IEEE Intelligent Systems **14** (1999) 20–26
15. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent Systems **16** (2001) 72–79
16. Zhong, N.: Ontologies in web intelligence. In: Practical Applications of Intelligent Agents, Springer. (2001)
17. Vianu, V.: A web odyssey: From codd to XML. In: ACM PODS Symposium on Principles of Database Systems. (2001) 1–15
18. Garofalakis, M., Rastogi, R., Seshadri, S., Shim, K.: Data mining and the web: Past, present and future. In: ACM CIKM'99 2nd Workshop on Web Information and Data Management (WIDM'99). (1999) 43–47
19. Chinenyanga, T., Kushmerick, N.: Expressive retrieval from XML documents. In: ACM SIGIR Conf on Research and Development in Information Retrieval. (2001) 163–171
20. Fan, W., Libkin, L.: On XML integrity constraints in the presence of DTDs. In: ACM PODS Symposium on Principles of Database Systems. (2001) 114–125
21. Fan, W., Simeon, J.: Integrity constraints for XML. In: ACM PODS Symposium on Principles of Database Systems. (2000) 23–34
22. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Intl Conf on Data Engineering (ICDE). (1995) 3–14
23. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: 5th Intl Conf on Extending Database Technology (EDBT). (1996) 3–17
24. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: PrefixSpan mining sequential patterns efficiently by prefix projected pattern growth. In: Intl Conf on Data Engineering (ICDE). (2001) 215–224
25. Srikant, R., Agrawal, R.: Mining generalized association rules. In: Intl Conf on Very Large Databases (VLDB). (1995) 407–419
26. Garofalakis, M., Rastogi, R., Shim, K.: SPIRIT: Sequential pattern mining with regular expression constraints. In: Intl Conf on Very Large Databases (VLDB). (1999) 223–234
27. Ng, R., Lakshmanan, L., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In: ACM SIGMOD Intl Conf on Management of Data. (1998) 13–24

28. Wang, K., Liu, H.: Discovering typical structures of documents: A road map approach. In: ACM SIGIR Intl Conf on Research and Development in Information Retrieval. (1998) 146–154
29. Wang, K., Liu, H.: Discovering structural association of semistructured data. Knowledge and Data Engineering **12** (2000) 353–371
30. Goldman, R., Widom, J.: DataGuides: Enabling query formulation and optimization in semistructured databases. In: Twenty-third Intl Conf on Very Large Data Bases. (1997) 436–445
31. Goldman, R., Widom, J.: Interactive query and search in semistructured databases. In: First Intl Workshop on the Web and Databases (WebDB). (1998) 52–62

Paraconsistent Query Answering Systems

Jørgen Villadsen

Informatics and Mathematical Modelling
Technical University of Denmark

Richard Petersens Plads, Building 321
DK-2800 Kongens Lyngby, Denmark

jv@imm.dtu.dk

Abstract. Classical logic predicts that everything (thus nothing useful at all) follows from inconsistency. A paraconsistent logic is a logic where inconsistency does not lead to such an explosion, and since in practice consistency is difficult to achieve there are many potential applications of paraconsistent logics in query answering systems. We compare the paraconsistent and the non-monotonic solutions to the problem of contradictions. We propose a many-valued paraconsistent logic based on a simple notion of indeterminacy. In particular we describe the semantics of the logic using key equalities for the logical operators. We relate our approach to works on bilattices. We also discuss and provide formalizations of two case studies, notably the well-known example involving penguins and a more interesting example in the domain of medicine.

Paraconsistent logic can be seen as an alternative, for example, to non-monotonic logic. Non-monotonists reject monotony because they think that there are experiences (most of the time involving birds) which show that monotony is wrong and in particular leads to some contradictions. But one who thinks the paraconsistent way would reject the principle of non contradiction and not monotony. The strategy of the paraconsistentist is more imaginative, he accepts to see penguins flying in the sky of Hawai's beaches and pink floyds surfing on Antarctica's permafrost. It seems to us that the future shall give the preference to paraconsistent logic taking in account the progress of genitical biology which already produces chicken without feathers, and in the future we may have flying pigs. In such an absurd world, it will make no sense to reason by default, because everything could be true by default.

J.-Y. Béziau
The Future of Paraconsistent Logic
Logical Studies Online Journal 2 (1999) p. 12

T. Andreasen et al. (Eds.): FQAS 2002, LNAI 2522, pp. 370–384, 2002.

1 Introduction

In symbolic logic the words model and proof are used in a different way than in mathematics and science in general and for this reason we avoid the somewhat ambiguous term “mathematical logic” [18]. For example, in relation to the science of biology the following statements can be said to be a model of a scenario involving a certain animal named Tweety (a well-known case study, cf. e.g. [5]).

Tweety is a penguin.	τ_0
Penguins are birds.	τ_1
Birds can fly.	τ_2
Penguins can not fly.	τ_3

As another and in many ways more substantial example we consider the following statements in the domain of medicine (adopted from a case study [8]).

John has symptom-1, symptom-3 and symptom-4 (but not symptom-2) and Mary has symptom-3 (but not symptom-1, symptom-2 and symptom-4).	κ_0
Disease-1 and disease-2 exclude each other.	κ_1
Symptom-1 and symptom-2 together imply disease-1.	κ_2
Symptom-1 and symptom-3 together imply disease-2.	κ_3
Symptom-1 and symptom-4 together imply disease-1.	κ_4
Symptom-3 implies disease-2 if symptom-1 is not present.	κ_5

Let τ be the set of statements τ_0, τ_1, τ_2 and τ_3; analogously for κ.

We call this kind of models not pertaining to symbolic logic mathematical models (even though there is no explicit use of numbers and related concepts of ordinary mathematics in the case studies τ and κ discussed here).

The overall aim of the present paper is to investigate the foundations of computer systems that can answer questions like the following.

Can Tweety fly?	Can Al fly?
Does John have disease-1?	Does Mary not have disease-2?

For simplicity we restrict ourselves to yes/no questions, but we believe that our results quite straightforwardly can be extended to many other query answering systems.

To be really useful query answering systems should not only answer the questions, but must, at least upon request, justify the answers given. We call such a justification a mathematical proof (it must be rigorous enough to satisfy even a professional mathematician, but of course it can differ in style and amount of details).

Let us focus on the Tweety example for a moment. One might argue that τ is not a mathematical model, since it is inconsistent; we can quickly produce mathematical proofs of two contradictory answers, θ and θ'.

Tweety can fly.	θ
Tweety can not fly.	θ'

To see what is going on it is beneficial to introduce another kind of models besides mathematical models. These are simply called models; we later introduce another kind of proofs as well.

The reason for regarding τ problematic is that τ has no models with respect to the semantics of classical logic. And classical logic is part of the foundations of mathematics even though the details of these issues appear to be rather unknown outside some circles of logicians, computer scientists and philosophers.

In logic semantics is the study of models [7]. The semantics of classical logic can be formulated in terms of two truth values: • (for truth) and ○ (for falsehood). If we use PT, BT and FT to mean that Tweety is a penguin, is a bird and can fly, respectively, then we initially have the following 8 models concerning Tweety without taking into account the statements τ.

Model #	PT	BT	FT
1	•	•	•
2	•	•	○
3	•	○	•
4	•	○	○
5	○	•	•
6	○	•	○
7	○	○	•
8	○	○	○

Using the semantics of classical logic, and without taking into account the statements τ, we can conclude neither θ nor θ', since it is not the case that, say, FT evaluates to the so-called designated truth value • in all models (see e.g. model # 8). But when we take into account the statements τ all models get eliminated and therefore any question we might possibly ask evaluates to the designated truth value • in all models *simply because there are no models with respect to the semantics of classical logic* (for τ and for κ as well).

The semantics of classical logic makes the logic explosive [6] in the sense that everything follows from an inconsistent set of statements like the case studies just presented. A logic where not everything follows from an inconsistency is said to be paraconsistent (inconsistency-tolerant).

In the table below we have indicated the answers that we find most appropriate for a paraconsistent logic — of course a query answering system might also in addition indicate the inconsistency.

		Classical Logic	Paraconsistent Logic
①	Is Tweety a penguin?	Yes	Yes
②	Is Tweety not a penguin?	Yes	No
③	Is Tweety a bird	Yes	Yes
④	Is Tweety not a bird?	Yes	No
⑤	Can Tweety fly?	Yes	Yes
⑥	Can Tweety not fly?	Yes	Yes
⑦	Can Al fly?	Yes	No
⑧	Can Al not fly?	Yes	No

The query answering system often has an inference engine that implements a particular proof system for the logic instead of working directly on the semantics. This kind of proofs must be distinguished from the mathematical proofs mentioned earlier because these proofs are completely formal in the sense that only syntactical manipulations using given axioms and inference rules are allowed (such proofs are almost never produced by professional mathematicians as they would be very large even with state of the art proof systems [17]).

For classical logic there are numerous proof systems [11], but they are all equivalent in the sense that they can be proved (using a mathematical proof) to be sound and (usually) complete with respect to the semantics of classical logic just outlined.

2 Aim and Overview

The aim of the present paper is to describe the symbolic logic ∇:

1. The logic ∇ is paraconsistent and can handle the case studies τ and κ.
2. The logic ∇ is based on a semantics that extends the one of classical logic.

We focus on the semantics of the logic ∇ since from the point of view of the users of the query answering system it is the most important aspect to grasp.

We have elsewhere discussed a proof system of the logic ∇ in the form of a sequent calculus [22]. The logic ∇ grew out of our recent work on the treatment of propositional attitudes like knowledge and belief in natural language [20]. Furthermore we are inspired by the use of the symbols Δ and ∇ in philosophical logic for determinacy and indeterminacy, respectively, cf. Evans [10] as the standard reference.

Our logic ∇ is a many-valued logic [14] and as such it differs from classical logic by the fundamental fact that it do not restrict the number of truth values to only two (to restrict the set of truth values of classical logic is meaningless, since at least two truth values are needed to distinguish truth from falsehood). Our approach is related to the four-valued logic of Belnap [4] who moves from truth values to partial truth values, namely sets of truth values.

In section 3 we briefly compare the paraconsistent and the non-monotonic approaches in relation to query answering systems and we introduce formalizations of the case studies. In section 4 we describe the notion of indeterminacy that we propose for our paraconsistent logic ∇ and in section 5 we elaborate on the two case studies. In section 6 we provide further details of related works. Finally section 7 comprises the conclusion.

3 Monotonicity

In the following we shall be more formal. We use the following binding priorities of the logical operators: negation $\neg$ (highest), conjunction $\wedge$, disjunction $\vee$ and implication $\rightarrow$ (lowest). This is a standard approach to reduce the number of parentheses in formulas.

We are also going to use the following abbreviations.

$$\Box\varphi \;\equiv\; \varphi = \top \qquad \varphi \rhd \psi \;\equiv\; \varphi \rightarrow \neg\psi$$
$$\varphi \lhd\!\rhd \psi \;\equiv\; (\varphi \rhd \psi) \wedge (\psi \rhd \varphi)$$

We use the symbol $\top$ for the designated truth value $\bullet$ (truth). Later we introduce the symbol $\bot$ for the non-designated truth value $\circ$ (falsehood).

We allow equality $=$ between formulas. In classical logic it corresponds to biimplication $\leftrightarrow$, but in our paraconsistent logic ∇ equality and biimplication differ (we provide truth tables later).

Also in classical logic the necessity operator $\Box$ as introduced above is vacuous (φ and $\Box\varphi$ are equivalent), but in ∇ we use $\Box\varphi$ to express that φ is classically true (and not contradictory). The operator $\Box$ is a so-called S5 modality [16].

The operators $\rhd$ and $\lhd\!\rhd$ are just conveniences for expressing that the first formula excludes and mutually excludes the second formula, respectively.

Reusing the labels for τ and κ we propose the following formalizations.

$$\Box PT \qquad \tau_0$$
$$\Box(Px \rightarrow Bx) \qquad \tau_1$$
$$Bx \rightarrow Fx \qquad \tau_2$$
$$Px \rhd Fx \qquad \tau_3$$

$$\Box(S_1J \wedge \neg S_2J \wedge S_3J \wedge S_4J \wedge \neg S_1M \wedge \neg S_2M \wedge S_3M \wedge \neg S_4M) \qquad \kappa_0$$
$$\Box(D_1x \lhd\!\rhd D_2x) \qquad \kappa_1$$
$$S_1x \wedge S_2x \;\rightarrow\; D_1x \qquad \kappa_2$$
$$S_1x \wedge S_3x \;\rightarrow\; D_2x \qquad \kappa_3$$
$$S_1x \wedge S_4x \;\rightarrow\; D_1x \qquad \kappa_4$$
$$\neg S_1x \wedge S_3x \;\rightarrow\; D_2x \qquad \kappa_5$$

We use D_i for disease-i, S_i for symptom-i, J for John and M for Mary.

Remember that $\rhd$ and $\lhd\!\rhd$ are just conveniences and that $\Box$ is vacuous in classical logic. The purpose for $\Box$ when we come to ∇ is primarily to simplify the analyses, but one can also argue that the observations τ_0 and κ_0, the taxonomic knowledge τ_1 and the "meta" knowledge κ_1 are considered to be classically true (and not contradictory).

One might argue that τ_2 is different — it speaks of *typical* birds (and penguins are not typical birds, because τ_3 shows at least one of their exceptional properties). Hence the implication $\rightarrow$ in τ_2 might be replaced by a so-called default implication leading to a non-monotonic logic [12] where even the following simple monotonicity rule must be abandoned.

If ψ follows from φ then ψ also follows from $\varphi \wedge \varphi'$ (for arbitrary ψ, φ, φ')

Take ψ to be FT, φ to be $\tau_0 \wedge \tau_1 \wedge \tau_2$ and φ' to be τ_3.

We consider non-monotonic logics to be a supplement rather than an alternative to paraconsistent logics. Since the issue of consistency is essential many proposals for both approaches exist and each have their pros and cons, but for the following reasons we find that non-monotonicity is an even more radical departure from classical logic than paraconsistency.

Semantics

As described we find it most advantageous that the logic behind query answering systems is based on a semantics. Classical logic is based on a well-defined semantics that is so intuitive that it is often not even explained. Unfortunately classical logic is explosive. However, the model elimination idea behind the semantics of classical logic seems to enforce monotonicity. Additional statements can only eliminate models and no conclusions can be retracted.

Typicality

In the τ case study we could speak of objects being typical or not, but in the κ case study we find it much harder to adopt such a distinction. Notice that there is no syntactic hint of why τ_2 should be a default implication; it all has to do with features of τ_1 and τ_3 as well. Even though one might propose ways to spot the typicality we find it worthwhile to investigate paraconsistency for query answering systems.

Modularity

Of course we could separate the statements about John and Mary completely (in κ_0^J and κ_0^M), but we would still be able to infer that John has, say, some other disease-3 (and doesn't have disease-3). Moreover, in order to have informative query answering systems we must combine information from various sources and of various categories and it is not clear to us how the notion of typicality behaves when modules are combined.

We realize that these arguments are far from conclusive.

One final point is that while it is often relatively easy to find and remove a contradiction in small case studies like τ and κ, it might be extremely difficult for larger theories of formalized common knowledge. The upcoming release 1.0 of the Cyc open source knowledge base (see the homepage OpenCyc.org) will consist of:

– 6,000 concepts: an upper ontology for all of human consensus reality.
– 60,000 assertions about the 6,000 concepts, interrelating them, constraining them, in effect (partially) defining them.

So we think a paraconsistent logic is needed and we now turn to our logic ∇.

4 Indeterminacy

The idea behind the paraconsistent logic ∇ is quite simple; besides the determinate truth values $\bullet$ and $\circ$ we add an indeterminate truth value $\shortmid$ (later we add additional indeterminate truth values). In the case study τ we then have the following model even though FT and $\neg FT$ holds (the model # is just a label).

Model #	PT	BT	FT
9	$\bullet$	$\bullet$	$\shortmid$

But the real work is to define suitable logical operators like implication $\rightarrow$ (it turns out that the usual definition $\varphi \rightarrow \psi \;\equiv\; \neg\varphi \vee \psi$ will not do).

The following semantic clauses are standard for classical logic (the clause for basic formulas is omitted and the superscript C indicates classical logic).

$$[\![\top]\!]^{\mathsf{C}} = \bullet$$

$$[\![\neg\phi]\!]^{\mathsf{C}} = \begin{cases} \bullet \text{ if not } [\![\phi]\!]^{\mathsf{C}} = \bullet \\ \circ \text{ otherwise} \end{cases}$$

$$[\![\phi \wedge \psi]\!]^{\mathsf{C}} = \begin{cases} \bullet \text{ if } [\![\phi]\!]^{\mathsf{C}} = \bullet \text{ and } [\![\psi]\!]^{\mathsf{C}} = \bullet \\ \circ \text{ otherwise} \end{cases}$$

$$[\![\phi \vee \psi]\!]^{\mathsf{C}} = \begin{cases} \bullet \text{ if } [\![\phi]\!]^{\mathsf{C}} = \bullet \text{ or } [\![\psi]\!]^{\mathsf{C}} = \bullet \\ \circ \text{ otherwise} \end{cases}$$

Notice that only the designated truth value $\bullet$ is mentioned outside the 'otherwise' case, since we simply explain the object language by means of the meta language. This is why we are almost forced to point out that the 'or' for $\vee$ is to be understood as "inclusive" and not "exclusive" as often in natural language — this can be avoided by the following semantic clause.

$$[\![\phi \vee \psi]\!]^{\mathsf{C}} = \begin{cases} \circ \text{ if } [\![\phi]\!]^{\mathsf{C}} = \circ \text{ and } [\![\psi]\!]^{\mathsf{C}} = \circ \\ \bullet \text{ otherwise} \end{cases}$$

Not only is the "inclusive/exclusive" ambiguity avoided, we also think that it is more natural to specify $\bullet$ for $\wedge$ and $\circ$ for $\vee$.

We easily avoid the 'not' for $\neg$ by the following semantic clause.

$$[\![\neg\phi]\!]^{\mathsf{C}} = \begin{cases} \bullet \text{ if } [\![\phi]\!]^{\mathsf{C}} = \circ \\ \circ \text{ otherwise} \end{cases}$$

The motivation for our logical operators is to be found in the key equalities shown to the right of the following semantic clauses (notice that even the ‘and’ for $\wedge$ is avoided).

$$\llbracket \top \rrbracket \;=\; \bullet$$

$$\llbracket \neg\varphi \rrbracket \;=\; \begin{cases} \bullet \quad \text{if } \llbracket\varphi\rrbracket = \circ \\ \circ \quad \text{if } \llbracket\varphi\rrbracket = \bullet \\ \llbracket\varphi\rrbracket \text{ otherwise} \end{cases} \qquad \begin{aligned} \top &= \neg\bot \\ \bot &= \neg\top \end{aligned}$$

$$\llbracket \varphi \wedge \psi \rrbracket \;=\; \begin{cases} \llbracket\varphi\rrbracket \text{ if } \llbracket\varphi\rrbracket = \llbracket\psi\rrbracket \\ \llbracket\psi\rrbracket \text{ if } \llbracket\varphi\rrbracket = \bullet \\ \llbracket\varphi\rrbracket \text{ if } \llbracket\psi\rrbracket = \bullet \\ \circ \quad \text{otherwise} \end{cases} \qquad \begin{aligned} \varphi &= (\varphi \wedge \varphi) \\ \psi &= (\top \wedge \psi) \\ \varphi &= (\varphi \wedge \top) \end{aligned}$$

In the semantic clauses several cases may apply if and only if they agree on the result. The semantic clauses work for classical logic and are taken as the starting point for ∇.

To put it in words: the motivation for $\neg$ is that $\top$ and $\bot$ swap, and the motivation for $\wedge$ is that the addition of something to itself does not give anything new (known as idempotency) and that $\top$ is neutral.

We use the following standard abbreviations.

$$\bot \;\equiv\; \neg\top \qquad \varphi \vee \psi \;\equiv\; \neg(\neg\varphi \wedge \neg\psi) \qquad \varphi \neq \psi \;\equiv\; \neg(\varphi = \psi)$$

We do not have $\varphi \vee \neg\varphi$ in ∇ since the indeterminate truth value $\shortmid$ is not designated (only $\bullet$ is designated). Since we obviously require that $\varphi \rightarrow \varphi$ the usual definition $\varphi \rightarrow \psi \;\equiv\; \neg\varphi \vee \psi$ will not do (cf. also [21]). Unfortunately it then follows that $(\varphi \wedge \neg\varphi) \rightarrow (\psi \vee \neg\psi)$ as can easily be verified using the truth tables below restricted to just $\bullet$, $\circ$ and $\shortmid$.

The reason for this problem is that in a sense there is not only a single indeterminacy, but a unique one for each basic formula. Hence it make sense to consider $\shortparallel$ as the alternative indeterminacy.

As discussed elsewhere [22] we think there should be a countable infinite number of truth values besides the two truth values of classical logic, but for the case studies it suffices to consider just these four truth values as we shall see in section 5.

Note that in ∇ the usual commutative, associative and distributive laws hold for $\wedge$ and $\vee$. Also the following De Morgan laws and the double negation law hold as in classical logic.

$$\neg(\varphi \wedge \psi) \;=\; (\neg\varphi \vee \neg\psi) \qquad \neg(\varphi \vee \psi) \;=\; (\neg\varphi \wedge \neg\psi) \qquad \neg\neg\varphi \;=\; \varphi$$

We have the following truth tables.

$\wedge$	●	○	∣	∥
●	●	○	∣	∥
○	○	○	○	○
∣	∣	○	∣	○
∥	∥	○	○	∥

$\vee$	●	○	∣	∥
●	●	●	●	●
○	●	○	∣	∥
∣	●	∣	∣	●
∥	●	∥	●	∥

$\neg$	
●	○
○	●
∣	∣
∥	∥

Observe that conjunction is just the identity function in the first row, in the first column and in the diagonal.

As for negation $\neg$ and conjunction $\wedge$ the motivation for the biimplication operator $\leftrightarrow$ (and hence the implication operator $\rightarrow$ as defined as an abbreviation) is based on the few key equalities shown to the right of the following semantic clauses.

$$[\![\varphi = \psi]\!] \;=\; \begin{cases} \bullet \text{ if } [\![\varphi]\!] = [\![\psi]\!] \\ \circ \text{ otherwise} \end{cases}$$

$$[\![\varphi \leftrightarrow \psi]\!] \;=\; \begin{cases} \bullet & \text{if } [\![\varphi]\!] = [\![\psi]\!] \\ [\![\psi]\!] & \text{if } [\![\varphi]\!] = \bullet \\ [\![\varphi]\!] & \text{if } [\![\psi]\!] = \bullet \\ [\![\neg\psi]\!] & \text{if } [\![\varphi]\!] = \circ \\ [\![\neg\varphi]\!] & \text{if } [\![\psi]\!] = \circ \\ \circ & \text{otherwise} \end{cases} \qquad \begin{array}{rcl} \top & = & (\varphi \leftrightarrow \varphi) \\ \psi & = & (\top \leftrightarrow \psi) \\ \varphi & = & (\varphi \leftrightarrow \top) \\ \neg\psi & = & (\bot \leftrightarrow \psi) \\ \neg\varphi & = & (\varphi \leftrightarrow \bot) \end{array}$$

As before several cases may apply if and only if they agree on the result and the semantic clauses work for classical logic too.

To put it in words: the motivation for $\leftrightarrow$ is that $\top$ is neutral and $\bot$ is linked to $\neg$, and the motivation for $\rightarrow$ is that something is implied if its addition does not give anything new.

$$\varphi \rightarrow \psi \;\equiv\; \varphi \leftrightarrow \varphi \wedge \psi$$

We have the following truth tables.

$\leftrightarrow$	●	○	∣	∥
●	●	○	∣	∥
○	○	●	∣	∥
∣	∣	∣	●	○
∥	∥	∥	○	●

$\rightarrow$	●	○	∣	∥
●	●	○	∣	∥
○	●	●	●	●
∣	●	∣	●	∣
∥	●	∥	∥	●

$=$	●	○	∣	∥
●	●	○	○	○
○	○	●	○	○
∣	○	○	●	○
∥	○	○	○	●

$\Box$	
●	●
○	○
∣	○
∥	○

We next show that negation $\neg$, conjunction $\wedge$ and equality $=$ suffice. We use Δ for determinacy and ∇ for indeterminacy with the following abbreviations.

$$\Delta\varphi \;\equiv\; \Box(\varphi \vee \neg\varphi) \qquad \nabla\varphi \;\equiv\; \neg\Delta\varphi$$

We also use the following auxiliary abbreviation.

$$\varphi \leadsto \psi \;\equiv\; \neg\Box\varphi \vee \psi$$

We have the following truth tables.

◁▷	●	○	ǀ	ǁ
●	○	●	ǀ	ǁ
○	●	●	●	●
ǀ	ǀ	●	●	○
ǁ	ǁ	●	○	●

▷	●	○	ǀ	ǁ
●	○	●	ǀ	ǁ
○	●	●	●	●
ǀ	ǀ	●	●	ǀ
ǁ	ǁ	●	ǁ	●

⇝	●	○	ǀ	ǁ
●	●	○	ǀ	ǁ
○	●	●	●	●
ǀ	●	●	●	●
ǁ	●	●	●	●

∇	
●	○
○	○
ǀ	●
ǁ	●

The central abbreviations for $\leftrightarrow$ is based directly on its semantic clause above.

$$\begin{aligned} \varphi \leftrightarrow \psi \;\equiv\; & (\varphi = \psi \rightsquigarrow \top) \wedge \\ & (\varphi \rightsquigarrow \psi) \wedge \\ & (\psi \rightsquigarrow \varphi) \wedge \\ & (\neg\varphi \rightsquigarrow \neg\psi) \wedge \\ & (\neg\psi \rightsquigarrow \neg\varphi) \wedge \\ & (\nabla\varphi \wedge \nabla\psi \wedge \varphi \neq \psi \rightsquigarrow \bot) \end{aligned}$$

Now all truth tables are calculated from the semantic clauses of negation $\neg$, conjunction $\wedge$ and equality $=$ (well, we also need $\top$).

5 Computing Answers

We now return to the case studies τ and κ introduced in section 1 and show how the answers to the yes/no questions can be computed in principle.

In section 3 we proposed the following formalization.

$\square PT$	τ_0
$\square(Px \rightarrow Bx)$	τ_1
$Bx \rightarrow Fx$	τ_2
$Px \triangleright Fx$	τ_3

Recall model # 9 where PT equals •, BT equals • and FT equals ǀ (there is also a model # 10 where FT equals ǁ, but it gives analogous results).

In order to deal with the questions ① – ⑧ listed in section 1 we use *FA* to mean that Al can fly and we get many more models (*FA* equals •, *FA* equals ∘, *FA* equals ǀ and *FA* equals ǁ). Let us assume that *FA* equals ∘ in model # 9.

In ∇ models either have the determinate truth value • or, in case a contradiction is present, an indeterminate truth value. As in classical logic we use the implication $\rightarrow$ to evaluate the questions (hence with the formula corresponding to τ on the left hand side and with the formula corresponding to the question on the right hand side) and see if it is the designated truth value •.

In classical logic this boils down to just looking at the right hand side for the models, but in ∇ there are more options even though as in classical logic only the truth value • is a designated truth value (since it does not make sense to have indeterminate truth values as designated truth values).

For example, the answer to question ① is yes, since PT equals $\bullet$ in model # 9 *and* all other models, and from the first column of the truth table for implication $\rightarrow$ we see that we have the designated truth value in all places (the $\circ$ row is not relevant, of course, since then we do not have a model).

The answer to question ② is no, since $\neg PT$ equals $\circ$ in model # 9, and from the second column of the truth table for implication $\rightarrow$ we see that we do not have the designated truth value in all places (well, we only have it for the $\circ$ row and as before it is not relevant).

The answer to both questions ⑤ and ⑥ is yes, since FT and $\neg FT$ both equal ǀ in model # 9, and from the third column of the truth table for implication $\rightarrow$ we see that we have the designated truth value in the ǀ row, *and* ditto for all other models (replacing ǀ with ǁ when necessary). Finally, the answer to question ⑦ is no, since FA equals $\circ$ in model # 9; same situation as for question ② considered above.

In section 3 we proposed the following formalization.

$$\Box(S_1J \land \neg S_2J \land S_3J \land S_4J \land \neg S_1M \land \neg S_2M \land S_3M \land \neg S_4M) \qquad \kappa_0$$

$$\Box(D_1x \mathrel{\triangleleft\!\triangleright} D_2x) \qquad \kappa_1$$

$$S_1x \land S_2x \;\rightarrow\; D_1x \qquad \kappa_2$$

$$S_1x \land S_3x \;\rightarrow\; D_2x \qquad \kappa_3$$

$$S_1x \land S_4x \;\rightarrow\; D_1x \qquad \kappa_4$$

$$\neg S_1x \land S_3x \;\rightarrow\; D_2x \qquad \kappa_5$$

We split κ into κ_J (x is J in κ) and κ_M (x is M in κ) and using the truth tables we get the following two intermediate tables that must then be combined.

D_1J	D_2J	κ_J	κ'_J	D_1M	D_2M	κ_M	κ'_M
●	●	○	○	●	●	○	○
●	○	○	○	●	○	○	○
●	ǀ	○	ǀ	●	ǀ	○	ǀ
●	ǁ	○	ǁ	●	ǁ	○	ǁ
○	●	○	○	○	●	●	●
○	○	○	○	○	○	○	○
○	ǀ	○	○	○	ǀ	ǀ	ǀ
○	ǁ	○	○	○	ǁ	ǁ	ǁ
ǀ	●	○	ǀ	ǀ	●	○	ǀ
ǀ	○	○	○	ǀ	○	○	○
ǀ	ǀ	ǀ	ǀ	ǀ	ǀ	ǀ	ǀ
ǀ	ǁ	○	○	ǀ	ǁ	○	○
ǁ	●	○	ǁ	ǁ	●	○	ǁ
ǁ	○	○	○	ǁ	○	○	○
ǁ	ǀ	○	○	ǁ	ǀ	○	○
ǁ	ǁ	ǁ	ǁ	ǁ	ǁ	ǁ	ǁ

The columns κ'_J and κ'_M correspond to the situation where $\Box$ is omitted from the rules (but not from the facts), hence replacing κ_1 with κ'_1.

$$D_1x \lhd\!\rhd D_2x \qquad\qquad \kappa'_1$$

Recall that κ is the set of statements κ_0, κ_1, κ_2 , κ_3 , κ_4 and κ_5. In the following $\kappa \vdash \varphi$ means that the answer to the question corresponding to the formula φ is yes (and $\kappa \not\vdash \varphi$ means that the answer is no). For example, $\kappa \vdash D_1J$ is a compact was to express that the query answering system should answer yes to the question: Does John have disease-1?

From the columns κ_J and κ_M we obtain:

$$\kappa \vdash D_1J \qquad \kappa \vdash \neg D_1J \qquad \kappa \vdash \neg D_1M \qquad \kappa \not\vdash D_1M$$

$$\kappa \vdash D_2J \qquad \kappa \vdash \neg D_2J \qquad \kappa \vdash D_2M \qquad \kappa \not\vdash D_2M$$

We consider here just the details of the first result, namely $\kappa \vdash D_1J$. For the combination κ we first observe that both columns κ_J and κ_M have ı and ıı rows, so κ can be ı and ıı (κ will never be • since κ_J is never •, cf. the truth table for the conjunction operator $\wedge$). We then observe that when κ is ı then D_1J is ı and using the truth table for the implication operator $\rightarrow$ we get • (the designated truth value). Similarly for ıı and hence we have $\kappa \vdash D_1J$.

We find these results the best possible (given that the information is classically inconsistent) because the inconsistency with respect to John does not lead to inconsistency with respect to Mary.

From the columns κ'_J and κ'_M we observe that if $\Box$ is omitted from κ_1 (see κ'_1), we obtain $\kappa \not\vdash D_1J$, $\kappa \not\vdash D_1M$ and $\kappa \not\vdash D_2J$, which is contrary to our intuition.

Instead of $\lhd\!\rhd$ we could consider Sheffer's stroke $|$ (where $\varphi|\psi$ is equivalent to $\neg(\varphi \wedge \psi)$), but $\Box(D_1x \,|\, D_2x)$ would not give any models — all rows are $\circ$ (the $\Box$ is needed for the same reason as in the $\lhd\!\rhd$ case just discussed). However, $\Box(D_1x \lhd\!\rhd D_2x \;\; \vee \;\; D_1x \,|\, D_2x)$ is an interesting combination where we obtain $\kappa \vdash \neg D_1J$ and $\kappa \vdash \neg D_2J$, but $\kappa \not\vdash D_1M$ since if D_1M equals ı and D_2M equals ıı we get ıı (and also if D_1M equals ıı and D_2M equals ı we get ı).

6 Related Works

Our logic ∇ is a generalization of Łukasiewicz's three-valued logic (originally proposed 1920–30), with the intermediate value duplicated many times and ordered such that none of the copies of this value imply other ones, but it differs from Łukasiewicz's many-valued logics [14]. However, rather than consider Łukasiewicz's logics we describe a four-valued logic that has become prominent more recently, namely the so-called bilattice semantics for the four-valued logic of Belnap [4].

In classical logic we have just two truth values, denoted True and False here. On the bilattice semantics we move from truth values to partial truth values, namely sets of truth values, as illustrated in the first column.

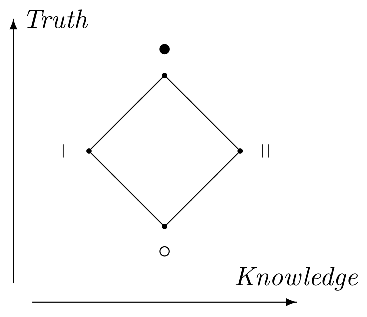
$\{\mathsf{True}\}$	**T**	$\top$	$\bullet$	Just true
$\{\mathsf{False}\}$	**F**	$\bot$	$\circ$	Just false
$\{\}$	**N**	$\dagger$	∣	Neither true nor false
$\{\mathsf{True}, \mathsf{False}\}$	**B**	$\ddagger$	∥	Both true and false

The second column shows the names used in the meta-language for the partial truth values in for example [14] and the fourth column shows the names that we have used. The third column shows the symbols that we use in our logic for the values (note that these are different from for example [14], but they correspond nicely to our names).

The idea is to have two partial orderings of these partial truth values:

- A *truth* lattice; with $\bullet$ at the top and $\circ$ at the bottom, and with ∣ and ∥ somehow "between", because they, in a suitable sense, if assigned to φ leaves open both possibilities that φ may "really" be true or be false.
- A *knowledge* lattice; with ∥ at the top as "over-determined" and ∣ at the bottom as "under-determined", and with $\bullet$ and $\circ$ somehow "between" corresponding to the classical truth values.

In the (Hasse) diagram below the *truth* lattice goes "bottom-up" and the *knowledge* lattice goes "left-to-right" (the two lattices are independent of one another, and together we have a bilattice [13]).



The corresponding *truth* lattice operations give the same conjunction (meet), disjunction (join) and negation operators (inversion) as in ∇. However, there are numerous choices for the implication and biimplication operators, but they all differ from the definitions used in ∇, cf. [14, pages 393–400] for a survey.

Even if we introduce the definitions for implications and biimplication operators used in ∇ in a logic based on the bilattice semantics, it would not work for the following reason. In classical logic as well as in our logic ∇ the truth value $\bullet$ is the only designated truth value, but on the bilattice semantics corresponding to the four-valued logic of Belnap there are not just one designated truth value, but two, namely: Just true & Both true and false — since the latter should be distinct from: Neither true nor false. If this distinction is omitted, as in the sequent calculus by Muskens [19], then we think that the logic is not really based on the notion of a bilattice but on the notion of indeterminacy like ∇.

7 Conclusion

In our opinion flexible query answering systems must deal with classically inconsistent information. Modern computational logic — concerned with all uses of logic in computer science — is one approach. In particular we would like to point out that we find the extensive literature on belief revision and update as well as on knowledge engineering techniques a supplement rather than an alternative to works on paraconsistent computational logic [9].

It is an important advantage of paraconsistency that it is compatible with monotonicity: Answers to queries remain correct no matter what additional information becomes available. In section 3 we described a number of problems for non-monotonic logics with respect to semantics, typicality and modularity.

Paraconsistent logic has found various applications in artificial intelligence, cf. [1], and in particular paraconsistent database systems have been proposed based on the bilattice semantics [2,3]. As described in section 6 our paraconsistent logic ∇ is different, but since both approaches use a four-valued logic we hope to use these works toward an implementation of a paraconsistent query answering system based on the indeterminacy semantics of ∇. The computations given in section 5 are of course very preliminary.

We have discussed the paraconsistent logic ∇ as a more or less propositional logic, but the results generalize even beyond first order logic to higher order logic and proof systems have been proposed [19,22] (albeit not implemented). A higher order logic can serve as a foundations of mathematics [15], but such theories are perhaps too powerful to be of use even in advanced knowledge bases at the moment. The paraconsistent logic ∇ is actually a relative small extension of classical logic to a four-valued logic, still with a single designated truth value and using key equalities for the logical operators. There are many alternatives to classical logic and further investigations as well as case studies are necessary in order to assess this approach to paraconsistent query answering systems.

References

1. J. M. Abe. Some recent applications of paraconsistent systems to AI. *Logique & Analyse*, 157:83–96, 1997.
2. R. Bagai. A query construct for paraconsistent databases. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, 428–434, 1998.
3. R. Bagai and R. Sunderraman. A paraconsistent relational data model. *International Journal of Computer Mathematics*, 55(1):39–55, 1995.
4. N. D. Belnap Jr. A useful four-valued logic. In J. M. Dunn and G. Epstein, editors, *Modern Uses of Multiple-Valued Logic*, pages 8–37. D. Reidel, 1977.
5. P. Besnard and A. Hunter. Introduction to actual and potential contradictions. In D. M. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems: Volume II*, pages 1–9. Kluwer, 1989.
6. J.-Y. Béziau. What is paraconsistent logic? In D. Batens et al., editors, *Frontiers in Paraconsistent Logic*, pages 95–111. Research Studies Press, 2000.
7. E. J. Borowski and J. M. Borwein. *Dictionary of Mathematics.* Collins, 1989.

8. N. C. A. da Costa and V. S. Subrahmanian. Paraconsistent logics as a formalism for reasoning about inconsistent knowledge bases. *Artificial Intelligence in Medicine*, 1:167–174, 1989.
9. Hendrik Decker, Jørgen Villadsen, and Toshiharu Waragai, editors. *Paraconsistent Computational Logic*, volume 95 of *Datalogiske Skrifter*, Roskilde, Denmark, July 27, 2002. Roskilde University.
10. G. Evans. Can there be vague objects? *Analysis*, 38(4):208, 1978.
11. D. Gabbay and F. Guenthner, editors. *Handbook of Philosophical Logic, Volume I: Elements of Classical Logic.* D. Reidel, 1983.
12. M. Ginsberg. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, 1987.
13. M. Ginsberg. Multivalued logics: A uniform approach to inference in artificial intelligence. *Computer Intelligence*, 4:265–316, 1988.
14. S. Gottwald. *A Treatise on Many-Valued Logics.* Research Studies Press, 2001.
15. W. S. Hatcher. *The Logical Foundations of Mathematics.* Pergamon Press, 1982.
16. G. E. Hughes and M. J. Cresswell. *An Introduction to Modal Logic.* Methuen, 1968.
17. D. A. MacKenzie. *Mechanizing Proof.* MIT Press, 2001.
18. E. Mendelson. *Introduction to Mathematical Logic.* Chapman, 4th edition, 1997.
19. R. Muskens. *Meaning and Partiality.* CSLI Publications, 1995.
20. J. Villadsen. Combinators for paraconsistent attitudes. In P. de Groote et al., editors, *Logical Aspects of Computational Linguistics*, pages 261–278. Springer, 2001. LNCS 2099.
21. J. Villadsen. Paraconsistent knowledge bases and many-valued logic. In H.-M. Haav et al. (editors), *International Baltic Conference on Databases and Information Systems*, Volume 2, pages 77–90, Tallinn, Estonia, 2002.
22. J. Villadsen. A paraconsistent higher order logic. In Decker et al. [9], pages 33–49. Updated PCL 2002 paper available at CoRR (Computing Research Repository) `http://arXiv.org/abs/cs.LO/0207088`

Author Index