Erhard Rahm (Ed.)

LNBI 2994

Data Integration in the Life Sciences

First International Workshop, DILS 2004
Leipzig, Germany, March 2004
Proceedings

Springer

Springer
Berlin
Heidelberg
NewYork
Hong Kong
London
Milan
Paris
Tokyo

Erhard Rahm (Ed.)

Data Integration in the Life Sciences

First International Workshop, DILS 2004
Leipzig, Germany, March 25-26, 2004
Proceedings

Springer

eBook ISBN: 3-540-24745-9
Print ISBN: 3-540-21300-7

Preface

DILS 2004 (Data Integration in the Life Sciences) is a new bioinformatics workshop focusing on topics related to data management and integration. It was motivated by the observation that new advances in life sciences, e.g., molecular biology, biodiversity, drug discovery and medical research, increasingly depend on bioinformatics methods to manage and analyze vast amounts of highly diverse data. Relevant data is typically distributed across many data sources on the Web and is often structured only to a limited extent. Despite new interoperability technologies such as XML and web services, integration of data is a highly difficult and still largely manual task, especially due to the high degree of semantic heterogeneity and varying data quality as well as specific application requirements.

The call for papers attracted many submissions on the workshop topics. After a careful reviewing process the international program committee accepted 13 long and 2 short papers which are included in this volume. They cover a wide spectrum of theoretical and practical issues including scientific/clinical workflows, ontologies, tools/systems, and integration techniques. DILS 2004 also featured two keynote presentations, by Dr. Thure Etzold (architect of the leading integration platform SRS, and president of Lion Bioscience, Cambridge, UK) and Prof. Dr. Svante Pääbo (Director, Max Planck Institute for Evolutionary Anthropology, Leipzig).

The workshop took place during March 25–26, 2004, in Leipzig, Germany, and was organized by the Interdisciplinary Bioinformatics Center (IZBI) of the University of Leipzig. IZBI was founded in 2002, and became one of five German bioinformatics centers funded by the German Research Association (DFG) after a highly competitive selection process. The University of Leipzig, founded in 1409, has chosen biotechnology as one of its high priority fields. Its new Center for Biotechnology and Biomedicine was established in 2003 and closely cooperates with IZBI and other research institutes as well as with industrial partners.

As the workshop chair and editor of this volume, I would like to thank all authors who submitted papers, as well as the program committee members and additional referees for their excellent work in evaluating the submissions. Special thanks go the Max Planck Institute for Evolutionary Anthropology for providing us with their new facilities to run the workshop, and the IZBI organization team, in particular Hai Hong Do, Toralf Kirsten and Hans Binder. Finally, I would like to thank Alfred Hofmann and his team at Springer-Verlag for their cooperation and help in putting this volume together.

Leipzig, February 2004

Erhard Rahm
Workshop Chair

International Workshop on Data Integration in the Life Sciences (DILS 2004)

Workshop Chair

Erhard Rahm	University of Leipzig, Germany

Program Committee

Howard Bilofsky	GlaxoSmithKline, USA
Terence Critchlow	Lawrence Livermore National Laboratory, USA
Peter Gray	University of Aberdeen, UK
Barbara Heller	University of Leipzig, Germany
Ralf Hofestaedt	University of Bielefeld, Germany
Jessie Kennedy	Napier University, Edinburgh, UK
Ulf Leser	Humboldt University, Berlin, Germany
Bertram Ludäscher	San Diego Supercomputer Center, USA
Sergey Melnik	Microsoft Research, USA
Peter Mork	University of Washington, Seattle, USA
Felix Naumann	Humboldt University, Berlin, Germany
Frank Olken	Lawrence Berkeley National Laboratory, USA
Norman Paton	University of Manchester, UK
Erhard Rahm	University of Leipzig, Germany
Louiqa Raschid	University of Maryland, USA
Kai-Uwe Sattler	Technical University Ilmenau, Germany
Steffen Schulze-Kremer	Freie Universität Berlin, Germany
Robert Stevens	University of Manchester, UK
Sharon Wang	IBM Life Sciences, USA
Limsoon Wong	Institute for Infocomm Research, Singapore

Additional Reviewers

Jens Bleiholder	Humboldt University, Berlin, Germany
Shawn Bowers	San Diego Supercomputer Center, USA
Hong-Hai Do	University of Leipzig, Germany
Ulrike Greiner	University of Leipzig, Germany
Efrat Jaeger	San Diego Supercomputer Center, USA
Kai Lin	San Diego Supercomputer Center, USA
Toralf Kirsten	University of Leipzig, Germany
Angela Stevens	Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
Melanie Weis	Humboldt University, Berlin, Germany

Sponsoring Institutions

Max Planck Institute for Evolutionary Anthropology, Leipzig, Germany
http://www.eva.mpg.de/

Interdisciplinary Center for Bioinformatics, University of Leipzig, Germany
http://www.izbi.de/

Organization Committee

Erhard Rahm	University of Leipzig, Germany
Hong-Hai Do	University of Leipzig, Germany
Toralf Kirsten	University of Leipzig, Germany
Hans Binder	University of Leipzig, Germany

Website

For more information on the workshop please visit the workshop website under the URL http://www.izbi.de/dils04/

Table of Contents

An Ontology-Driven Framework for Data Transformation in Scientific Workflows*

Shawn Bowers and Bertram Ludäscher

San Diego Supercomputer Center
University of California, San Diego
La Jolla, CA, 92093-0505, USA,
{bowers, ludaesch}@sdsc.edu

Abstract. Ecologists spend considerable effort integrating heterogeneous data for statistical analyses and simulations, for example, to run and test predictive models. Our research is focused on reducing this effort by providing data integration and transformation tools, allowing researchers to focus on "real science," that is, discovering new knowledge through analysis and modeling. This paper defines a generic framework for transforming heterogeneous data within scientific workflows. Our approach relies on a formalized ontology, which serves as a simple, unstructured global schema. In the framework, inputs and outputs of services within scientific workflows can have structural types and separate semantic types (expressions of the target ontology). In addition, a *registration mapping* can be defined to relate input and output structural types to their corresponding semantic types. Using registration mappings, appropriate data transformations can then be generated for each desired service composition. Here, we describe our proposed framework and an initial implementation for services that consume and produce XML data.

1 Introduction

For most ecological and biodiversity forecasting, scientists repeatedly perform the same process: They select existing datasets relevant to their current study, and then use these datasets as input to a series of analytical steps (that is, a *scientific workflow*). However, ecological and biodiversity data is typically heterogeneous. Researchers must spend considerable effort integrating and synthesizing data so that it can be used in a scientific workflow. Reusing analytical steps within workflows involves a similar integration effort. Each analytic step (or *service* since they can be implemented as web services [CCMW01]) in a workflow consumes and produces data with a particular structural representation, much like a dataset. To compose existing services, the structural and semantic differences between the services must be resolved, and this resolution is typically performed by the scientist either manually or by writing a special-purpose program or script.

* This work supported in part by the National Science Foundation (NSF) grants ITR 0225676 (SEEK) and ITR 0225673 (GEON), and by DOE grant DE-FC02-01ER25486 (SciDAC-SDM).

The Science Environment for Ecological Knowledge (SEEK)[1] [Mic03] is a multidisciplinary effort aimed at helping scientists discover, access, integrate, and analyze distributed ecological information. We envision the use of web-enabled repositories to store and access datasets, including raw data and derived results, software components, and scientific workflows. Additionally, we wish to exploit formalized, ecological ontologies to help scientists discover and integrate datasets and services, as workflows are designed and executed.

This paper proposes a framework that exploits ontological information to support structural data transformation for scientific workflow composition. We believe data transformation is an integral part of *semantic mediation,* which aims at providing automated integration services within SEEK. The framework is designed to allow researchers to easily construct scientific workflows from existing services, without having to focus on detailed, structural differences. In particular, when a service is stored in SEEK, each input and output is annotated with a structural and (optionally) a semantic type. In our framework, a structural type—similar to a conventional programming-language data type—defines the allowable data values for an input or output, whereas a semantic type describes the high-level, conceptual information of an input or output, and is expressed in terms of the concepts and properties of an ontology. Thus, although structurally different, two services may still be semantically compatible based on their semantic types.

The goal of the framework is to exploit semantic types to (semi-) automatically generate mappings between services with heterogeneous structural types. By defining input and output *registration mappings,* which link a structural type to a corresponding semantic type, ontological information is used to inform data transformation. When a scientist wishes to compose two services, the input and output registration mappings are combined to create a correspondence between the two structural types. The correspondence is then used, when possible, to generate the desired data transformation, thus making the services structurally compatible.

The rest of this paper is organized as follows. Section 2 briefly describes scientific workflows and introduces an example workflow used throughout the paper. Section 3 defines our proposed framework for data transformation. Section 4 describes an initial implementation of the framework for services that exchange XML data. In particular, we define a language for specifying registration mappings, based on structural types expressed using XML Schema. Section 5 discusses related work. Section 6 concludes with a discussion of future work.

2 Scientific Workflows

SEEK extends the Ptolemy system [BCD+02] to support scientific workflows. Ptolemy is an open-source, Java-based application that provides interfaces for designing and executing data-flow process networks [LP95]. In particular, we

[1] See http://seek.ecoinformatics.org/

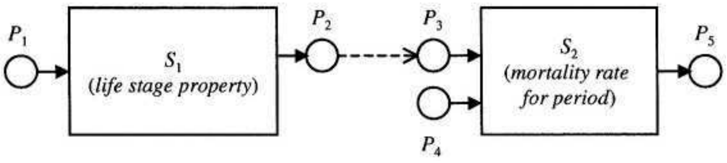


Fig. 1. Two connected services to calculate the k-value [BHT96] during a particular life-stage period.

are extending Ptolemy to support web-services, web-enabled repositories and workflow execution, scientific datasets, and semantic-type annotations through ontologies. This section briefly describes scientific workflows and introduces our running example. We note that our definition of a scientific workflow is inspired by, and compatible with, the dataflow models of Ptolemy.

We define a *scientific workflow* as a set of connected services. By *service,* we mean any software component that takes input and produces output, including but not limited to web services. A service has zero or more uniquely named *ports.* A port serves as either an input or an output to a service. Services exchange information using *connections,* which link an output port to one ore more input ports. When a pipeline is executed, data is transferred via connections from output to input ports according to an execution model. An execution model is an algorithm that determines how services should be scheduled and how data should flow through services.

Figure 1 depicts a simple workflow that contains two services S_1 and S_2, where the output port P_2 of S_1 is connected to the input port P_3 of S_2. The purpose of this simple workflow is to compute mortality rates for periods within the lifecycle of an organism [BHT96]. For example, consider the two datasets shown in Figure 2. The table on the left, Figure 2(a), gives population samples for specific development phases of the common field grasshopper (taken from Begon, *et al* [BHT96]). The dataset on the right, Figure 2(b), gives periods of development defined in terms of phases. Note that only one such period is given. Here, S_2 applies the "killing power," or k-value statistic to determine the rate of mortality for a set of observations (given in P_3) and a set of phases (given in P_4). For the datasets in Figure 2, S_2 would output a single pair (Nymphyl, 0.44) on port P_5. We note that, in SEEK, S_1 and S_2 represent stand-alone services that would be selected by an ecologist from a repository and connected, possibly as part of a larger workflow.

We require each port to have a *structural type,* which is similar to a conventional data type found in programming languages. In general, a structural type is expressed according to a *type system,* which consists of a set of base types, a description of the allowable types (of the type system), and rules defining when one type is a subtype of another. A type definition is a restriction on the structure of values being produced or consumed by a port. Thus, any value

(a)

Phase	*Observed*
Eggs	44,000
Instar I	3,513
Instar II	2,529
Instar III	1,922
Instar IV	1,461
Adults	1,300

(b)

Period	*Phases*
Nymphal	{Instar I, Instar II, Instar III, Instar IV}

Fig. 2. Example datasets for computing k-values during lifecycle periods.

that conforms to the structural type (or a subtype of the structural type) is an allowed value for the port.

We assume the function *structType(P)* is well defined for each port P and returns the structural type of P. Given two ports P_a and P_b, we write $P_a \preceq P_b$ to denote that the structural type of P_a is a subtype of the structural type of P_b. If $P_a \preceq P_b$, we say P_a is *structurally compatible* with P_b.

One of the goals of SEEK is to allow scientists to reuse existing services when building new ecological models. In general, we assume services are not "designed to fit." That is, two distinct services may produce and consume compatible information, but have incompatible structural types.

3 Ontologies for Data Transformation

In this section, we define our proposed framework for ontology-driven data transformation. We first describe the use of semantic-type annotations in SEEK for enriching workflow services. We then describe how semantic types are exploited in our framework via registration mappings, which are used to generate data transformations between structurally incompatible services.

3.1 Ontologies and Services

As part of SEEK, we are developing technology that lets ecologists define, manage, and exploit ontologies. In general, we permit scientists to construct domain or project-specific ontologies and define mappings between them, when appropriate. In addition, we are developing, with the help of ecologists, an upper-level ecological ontology to serve as a framework for incorporating the various domain-specific ontologies. The upper-level ontology includes concepts and relationships for ecological properties, methods, measurements, and taxonomies. In the rest of this paper, we assume a single global ontology, however, in our envisioned environment there will more likely be many ontologies that when combined, for example, through mappings, form a single, global ontology.

A *semantic type* is defined using the concepts and properties of the ontology. In SEEK, we use semantic types to annotate services and datasets, where a semantic-type annotation defines the conceptual information represented by the

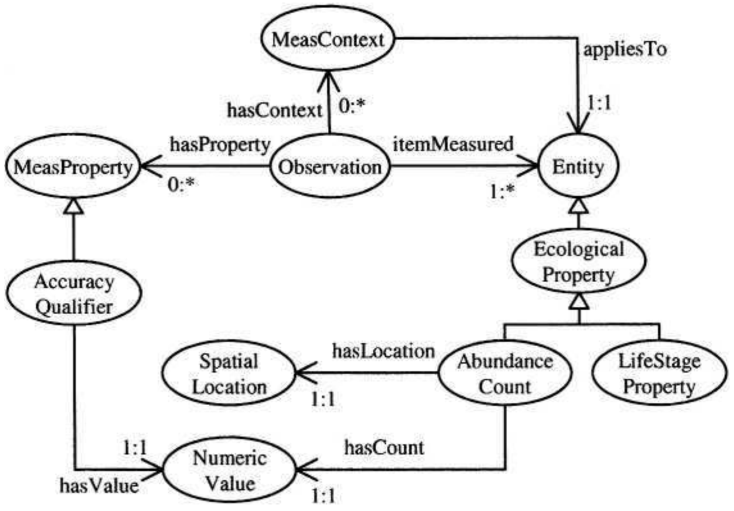


Fig. 3. Portion of a SEEK ontology for ecological measurements.

item. A semantic type defines the conceptual information that is either consumed or produced by a port. Thus, semantic types for ports are similar to semantic pre- and post-conditions in DAML-S [ABH+02]. We note that in SEEK, however, semantic types are independent of the structural details of a port. We believe decoupling structural and semantic types has the following advantages. First, although we require structural types, semantic types are optional. Thus, a service can still be used even if it is not fully specified. Second, a port's semantic type can be defined or updated after the service is deployed, without requiring changes to the structural type. Finally, we believe semantic types are easier to specify if they do not mix structural information. For example, a port's semantic type may be as simple as a single concept label, even though it has a complex structural type.

We have adopted the Web Ontology Language (OWL) [MvH03] to express ontologies in SEEK. Figure 3 shows a fragment of the SEEK measurement ontology. The ontology is represented graphically, using RDF-Schema conventions [BG03]. Only a subset of the OWL constructs are used in Figure 3. In particular, we only consider class and property definitions, subclass relationships, and cardinality constraints on properties. (The notation 0:* and 1:* represent value and existential restrictions, respectively, and 1:1 represents an exactly-one number restriction [BN03].) Similarly, Figure 4 gives the semantic types for ports P_2 and P_3 of Figure 1. As shown, the semantic type of port P_3 accepts observations, each of which measures an abundance count within the context of a life-stage property. The semantic type of P_2 outputs similar observations, but with accuracy qualifiers. In general, the semantic type of a port is defined as an OWL concept.

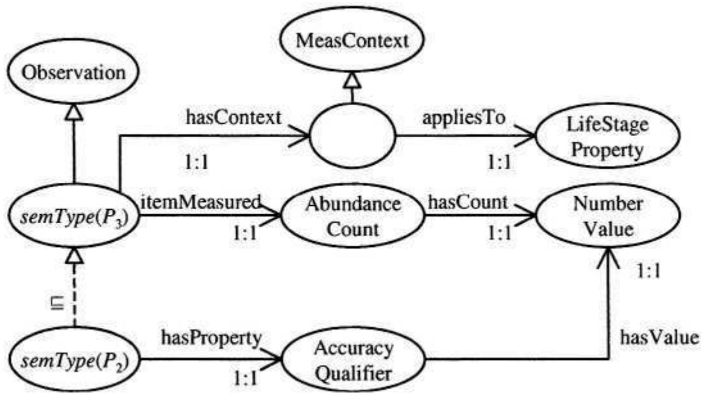


Fig. 4. Example semantic types for output and input ports P_2 and P_3.

We assume the function *semType(P)* returns the semantic type of a port *P*. As stated above, a semantic type denotes a concept, which either exists as, or is a restricted subclass of, a concept within an ontology. As shown in Figure 4, $semType(P_2)$ is a subtype of $semType(P_3)$, which we denote using the standard $\sqsubseteq$ relation used in description logics. Intuitively, $P_2 \sqsubseteq P_3$ holds if every instance of the concept $semType(P_2)$ is also an instance of the concept $semType(P_3)$.

3.2 Connecting Semantically Compatible Services

To correctly compose two services, a valid connection must be defined between an output port of the source service and an input port of the target service. Intuitively, a desired connection (see Figure 5) is valid if it is both semantically and structurally valid. A connection from port P_s to port P_t is *semantically valid* if $P_s \sqsubseteq P_t$ and *structurally valid* if $P_s \preceq P_t$. The connection is *structurally feasible* if there exists a structural transformation δ such that $\delta(P_s) \preceq P_t$.

Our focus here is on the situation shown in Figure 5. Namely, that we have semantically valid connections that are not structurally valid, but feasible. Thus, the goal is to find a δ that implements the desired structural transformation.

3.3 Structural Transformation Using Registration Mappings

Figure 6 shows our proposed framework for data transformation in scientific workflows. The framework uses *registration mappings,* which consists of a set of rules that define associations between a port's structural and semantic types. In this paper, we use registration mappings to derive data transformations. A rule $q \leftrightarrow p$ in a registration mapping associates data objects identified with a query q to concepts identified with a concept expression p. For example, q may expressed as an XQuery [BCF+03] for XML sources or as an SQL query for relational sources, selecting a set of objects belonging to the same concept

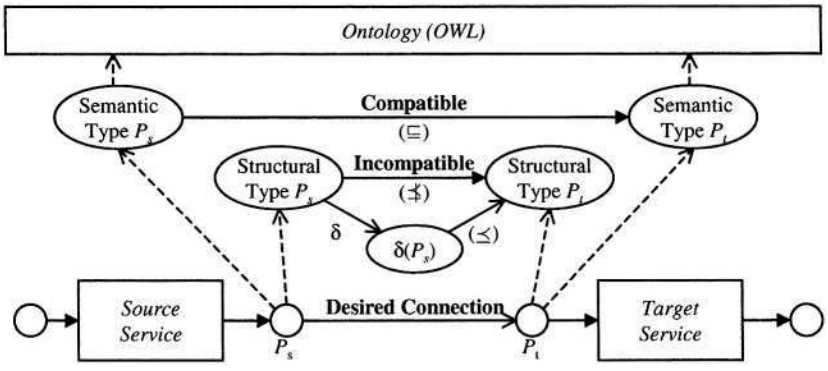


Fig. 5. A semantically valid, but structurally invalid connection.

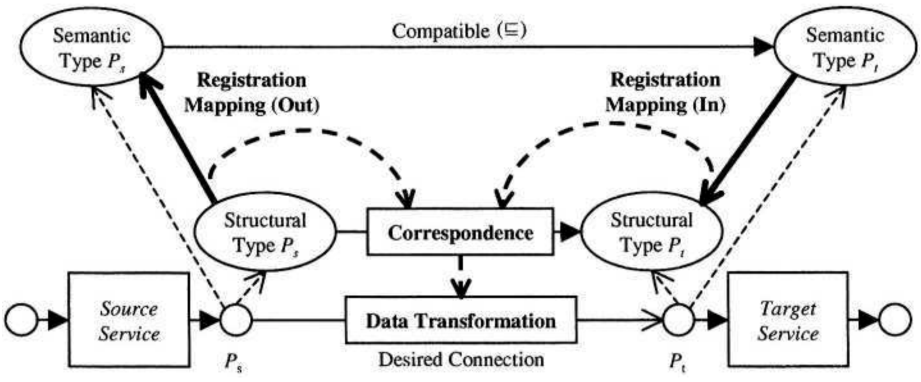


Fig. 6. The proposed transformation framework.

expression p. Thus, $q \leftrightarrow p$ is the part of a registration mapping that registers the q-selected objects with concepts denoted by p. We note that in general, p can represent a single concept or possibly a complex, description logic expression. In this section, we describe the general approach shown in Figure 6, and in the next section we define a specific implementation of the framework for XML sources, which includes a specific language for expressing registration mappings.

A registration mapping consists of one or more rules of the form $q \leftrightarrow p$. The expression q is a query that selects instances of the structural type to register to a concept denoted by p from the semantic type. We call the query q a *sub-structure* selection. In this paper, we only consider concept expressions p that denote *contextual paths* as opposed to arbitrary description logic expressions. A context path denotes a concept, possibly within the context of other concepts. For example, using the ontology fragment given in Figure 3, *Observation. itemMeasured.hasLocation* is a contextual path, where the concept selected by the path is *SpatialLocation* within the context of an observation measurement. Given a registration mapping rule $q \leftrightarrow$*Observation.itemMeasured.hasLocation,* we say that each data object selected by q is a *SpatialLocation* object for an *AbundanceCount,* which is the item being measured for some *Observation* object. It is important to note that we do not explicitly provide unique identifiers for data objects within a mapping rule. Instead, we use the contextual path simply as an annotation mechanism. Thus, data objects are identified locally to a structural-type instance, and these local identifiers are annotated with the appropriate semantic types. When a contextual-path expression is defined, it can be checked for validity (that is, that it makes sense) with respect to the semantic type, for example, with the help of a description-logic reasoner..

We distinguish between *output* and *input* registration mappings, where an output mapping is a registration mapping defined for an output port, and an input mapping is a registration mapping defined for an input port. Output and input registration mapping rules are composed to construct a *correspondence mapping* between the source and target structural types (see figure below). A correspondence mapping consists of rules of the form $q_s \mapsto q_t$, where q_s and q_t are queries against instances of output (source) and input (target) structural types, respectively. A rule in a correspondence mapping states an association between the two sub-structures defined by q_s and q_t. We construct individual correspondence mapping rules by composing registration mapping rules as shown in the figure below. Note that from a correspondence mapping, we wish to construct the appropriate transformation function δ.

$$\begin{tikzcd}
p_s \arrow[r, "\sqsubseteq_{path}"] & p_t \arrow[d] \\
q_s \arrow[u] \arrow[r, dashed, "\delta"'] & q_t
\end{tikzcd}$$

Given a set of output registration-mapping rules R_s and a set of input registration-mapping rules R_t, the correspondence mapping M between R_s and R_t is exactly the result of the *semantic join* of R_s with R_t (written $R_s \bowtie_{sem} R_t$), which we defined as follows. If $q_s \leftrightarrow p_1 \in R_s$ and $q_t \leftrightarrow p_2 \in R_t$, then $q_s \mapsto q_t \in R_s \bowtie_{sem} R_t$ if and only if $p_1 \sqsubseteq_{path} p_2$. The expression $p_1 \sqsubseteq_{path} p_2$ holds if p_1 is a *contextual subpath* of p_2, that is, if the concept denoted by p_1 is a subconcept of the concept denoted by p_2, and the context of p_1 is a subcontext of the context of p_2. We formally define the $\sqsubseteq_{path}$ relation for a simple semantic-path language in the next section.

As shown in Figure 6, we use a set of correspondence mappings M to generate a data transformation that can convert valid source data to valid target data. We note that correspondence mappings may be *underspecified,* that is, there may be more than one possible data transformation based on the correspondence mapping.

In general, we require registration mappings to be well-formed, that is, each query and contextual path is well-formed. We also make the following assumptions concerning registration mappings. We note that the complexity of enforcing the following properties depends on the languages used to express structural types, structural queries, semantic types, and contextual paths.

– **Consistent with respect to cardinality constraints**. A registration mapping is consistent with respect to cardinality constraints if, based on the rules of the mapping, the structural-type cardinality constraints imply the semantic-type cardinality (i.e., value restriction) constraints. Note that if the cardinality constraints between the structural and semantic types are inconsistent, the generated data transformation program may not be correct.
– **Partially complete**. A registration mapping is considered *structurally* complete if every data item required by a structural type is registered with some contextual path in the semantic type. Similarly, a registration mapping is *semantically* complete if every required concept in a semantic type (according to cardinality constraints) is registered by some data item in an instance of the structural type. We require all input registration mappings to be structurally complete. We also require all output registration mappings to be semantically complete with respect to the input registration mapping. That is, the output registration mapping should map data items to the needed concepts of the input registration mapping.

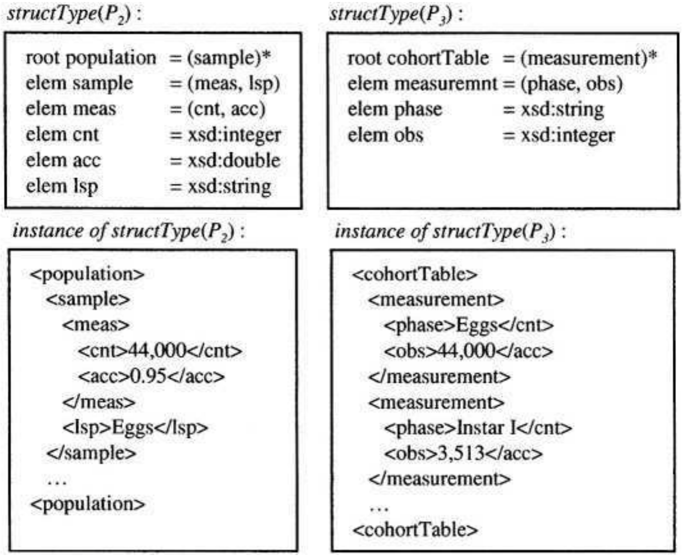
4 A Framework Implementation for XML-based Services

In this section, we describe an instantiation of the framework defined in Section 3 in which XML is used to interchange data between services. We define a subset of XML Schema [BM01, TBMM01] for the structural-type system, a simple XPath-based query language, a contextual-path language, and the contextual subpath relationship for contextual paths.

4.1 XML-based Structural Types

We consider a subset of XML Schema for expressing structural types, which is very close to Document Type Definitions (DTDs), but with XML datatypes [BM01]. In particular, a structural-type consists of one or more element definitions, one of which is a distinguished root element. For simplicity, we do not consider attributes. Instead, we use the common assumption that attributes are modeled as simple elements containing text values (where attribute names are often prefixed with the '@' symbol).

A structural type takes the form `root` e=cm or `elem` e=cm, where e is an element tag name and cm is the content model for e. Every structural type has exactly one root-element definition (which is distinguished by the `root` qualifier). A structural type may also have any number of additional element definitions (which are distinguished by the `elem` qualifier).



structType(P_2) :

```
root population  = (sample)*
elem sample      = (meas, lsp)
elem meas        = (cnt, acc)
elem cnt         = xsd:integer
elem acc         = xsd:double
elem lsp         = xsd:string
```

structType(P_3) :

```
root cohortTable  = (measurement)*
elem measuremnt = (phase, obs)
elem phase          = xsd:string
elem obs            = xsd:integer
```

instance of structType(P_2) :

```
<population>
  <sample>
    <meas>
      <cnt>44,000</cnt>
      <acc>0.95</acc>
    </meas>
    <lsp>Eggs</lsp>
  </sample>
  ...
<population>
```

instance of structType(P_3) :

```
<cohortTable>
  <measurement>
    <phase>Eggs</cnt>
    <obs>44,000</acc>
  </measurement>
  <measurement>
    <phase>Instar I</cnt>
    <obs>3,513</acc>
  </measurement>
  ...
<cohortTable>
```

Fig. 7. Example structural types and instances for port P_2 (left) and port P_3 (right).

We permit datatype and regular-expression content models. For example, the expression `elem cnt = xsd: integer` restricts `cnt` elements to contain integer values (where `xsd:integer` is an XML-Schema datatype). A content model is defined using standard DTD regular-expression syntax. For example, the element definition `elem sample = (meas,lsp)` states that a `sample` element consists of a `meas` element followed by a `lsp` element. We require each element definition to be nested under the root element either directly, as a subelement, or indirectly, through any number of intermediate subelements.

To illustrate, we assume that ports P_2 and P_3 in Figure 1 have the structural-type definitions given in the top of Figure 7. The structural type of P_2 is shown on the top-left and the structural type of P_3 is shown on the top-right. Sample instances (XML documents) are given below each corresponding structural type. As shown, the structural types are not subtypes, that is, $P_2 \not\preceq P_3$. In general, subtyping in XML is based on datatype compatibility (for example, `xsd:negativeInteger` is a subtype of `xsd:integer`) and various rules concerning content models (for example, the content model `(a,b,c)` is a subtype of the content model `(a|b|c)*` for element definitions `a`, `b`, and `c`).

4.2 Structural-Type Queries and Contextual Paths

We define a structural-type query q_{xp} for XML using a subset of XPath [CD99]. A query q_{xp} is expressed using the following syntax. We note that the fragment of XPath we consider is similar to *tree patterns* [MS02].

$$\begin{aligned} q_{xp} &= /p \\ p &= n \mid p/p \mid p/\texttt{text()} \mid p[q] \\ q &= p \mid p{=}v \end{aligned}$$

As shown, n represents an element tag name and v represents a text value (that is, PCDATA). The expression `/text()` selects the PCDATA content of an element.

A query is expressed as a simple path that always starts from the root element type defined in the XML structural type. A query returns either fragments (that is, sub-trees) of a document, or using the `text()` operator, a set of data values. In addition, we permit simplequalifiers q, which select sub-trees that contain a particular set of subelements, possibly with a specific data value.

The language we consider for expressing contextual paths is defined as follows. A contextual path defined over port P takes the form $semType(P).r_1.r_2...r_n$ for $n \geq 0$, where r_1 to r_n are valid properties defined for the semantic type of P. A contextual path always begins from the semantic type of a port. Additionally, a contextual path always denotes a single concept, which is either the semantic type of the port or a concept related to the semantic type through properties r_1 to r_n. Given a contextual path p, the function $leadsTo(p)$ returns the concept denoted by the contextual path p.

4.3 Registration and Correspondence Mappings

Figure 8 gives an example input and output registration mapping for ports P_2 and P_3 of Figure 6 (using the structural types for P_2 and P_3 defined in Figure 7). Assuming we wish to connect port P_2 to port P_3, we first use the input registration mapping R_s and the output registration mapping R_t of Figure 8 to generate a correspondence mapping. Recall that a correspondence mapping M for R_s and R_t is equivalent to $R_s \bowtie_{sem} R_t$, which is computed using the semantic subpath relation $\sqsubseteq_{path}$ (see Section 3.3).

We define the semantic subpath relation for our contextual path language as follows. Let p_1 and p_2 be the following contextual paths expressed over ports P_a and P_b, respectively.

$$\begin{aligned} p_1 &= semType(P_a).r_{11}.r_{12}. \; ... \; .r_{1n} \\ p_2 &= semType(P_b).r_{21}.r_{22}. \; ... \; .r_{2m}. \end{aligned}$$

The relation $p_1 \sqsubseteq_{path} p_2$ is true if and only if $m = n$, $P_a \sqsubseteq P_b$, and for $1 \leq i \leq n$, $r_{1i} = r_{2i}$ and $leadsTo(semType(P_a).r_{11}. \; ... \; .r_{1i}) \sqsubseteq leadsTo(semType(P_b).r_{21}. \; ... \; .r_{2i})$. To illustrate, the correspondence mapping M that results from applying the semantic join of R_s and R_t (from Figure 8) is shown in Figure 9.

4.4 Generating Data-Transformation Programs

In general, we want to use correspondence mappings to construct valid target data instances (that is, XML documents that conform to the target structural

P_2 output registration-mapping rules ($q_s \leftrightarrow p$):

`/population/sample`	$\leftrightarrow$ *semType*(P_2)
`/population/sample/meas/cnt`	$\leftrightarrow$ *semType*(P_2).itemMeasured
`/population/sample/meas/cnt/text()`	$\leftrightarrow$ *semType*(P_2).itemMeasured.hasCount
`/population/sample/meas/acc`	$\leftrightarrow$ *semType*($P2$).hasProperty
`/population/sample/meas/acc/text()`	$\leftrightarrow$ *semType*($P2$).hasProperty.hasValue
`/population/sample/lsp/text()`	$\leftrightarrow$ *semType*($P2$).hasContext.appliesTo

P_3 input registration-mapping rules ($p \leftrightarrow q_t$):

`/cohortTable/measurement`	$\leftrightarrow$ *semType*($P3$)
`/cohortTable/measurement/obs`	$\leftrightarrow$ *semType*($P3$).itemMeasured
`/cohortTable/measurement/obs/text()`	$\leftrightarrow$ *semType*($P3$).itemMeasured.hasCount
`/cohortTable/measurement/phase/text()`	$\leftrightarrow$ *semType*($P3$).hasContext.appliesTo

Fig. 8. Example registration mappings for ports P_2 and P_3.

`/population/sample`	$\mapsto$ `/cohortTable/measurement`
`/population/sample/meas/cnt`	$\mapsto$ `/cohortTable/measurement/obs`
`/population/sample/meas/cnt/text()`	$\mapsto$ `/cohortTable/measurement/obs/text()`
`/population/sample/lsp/text()`	$\mapsto$ `/cohortTable/measurement/phase/text()`

Fig. 9. Correspondence mapping that results from connecting ports P_2 and P_3.

type) from valid source data instances. The correspondence mapping from Figure 9 provides the following guidelines for developing such a mapping. Each population sample (`/population/sample`) should be used to generate a cohort table measurement (`/cohortTable/measurement`), where each sample's measurement count (`meas/cnt`) corresponds to an observation in the measurement element of the cohort table (`obs`), and each sample's life stage property (`lsp/text()`) corresponds to a phase in the measurement element of the cohort table (`phase`). Based on the correspondence mapping, the following data transformation (expressed using XQuery [BCF+03]) should be generated.

```
<cohortTable>
   { for $s in /population/sample return
         <measurement>
            { for $c in $s/meas/cnt return <obs>{$c/text()}</obs> }
            { for $l in $s/lsp return <phase>{$l/text()}</phase> }
         </measurement> }
</cohortTable>
```

To generate the above data transformation, we must make the following assumptions.

1. **Common prefixes refer to the same element.** We assume correspondence mapping rules that share a common prefix (in the source or target

side of the rule) refer to the same element (in the source or target side, respectively). Note that all correspondence rules, by definition, share at least the root element. For example, the first two correspondence rules in Figure 9 share the common source prefix /`population/sample` and target prefix /`cohortTable/measurement`. Thus, for a /`population/sample` element s mapped to a /`cohortTable/measurement` element m, we assume that each `cnt` element c under s is mapped to an `obs` element o under m.

2. **Correspondence rules have compatible cardinality restrictions.** By cardinality restrictions for XML, we mean whether a subelement can occur zero or more, one or more, zero or one, or exactly once under an element. For example, the XML fragments selected by the source and target side of the first correspondence rule in Figure 9 have identical cardinality restrictions. Namely, a root `population` element can contain zero or more `sample` elements and similarly, a root `cohortTable` element can contain zero or more `measurement` elements. In general, the source restrictions should be equivalent or stricter than the target restrictions. Thus, we do not want to map multiple source elements to a single target element.
3. **Datatypes are compatible.** We assume that each correspondence mapping rule between XML text values is valid. In particular, we assume that the datatype of the source is a subtype of the target. For example, if `cnt` was defined as a double type instead of an integer type, the third correspondence mapping rule would violate our subtype assumption, because a double is not a subtype of an integer (where `obs` requires integer content).

For registration mappings that satisfy the above assumptions and are known to be partially complete (see Section 3.3), it is straightforward to generate the appropriate data-transformation program. However, for mappings that are not partially complete (as given in Section 3.3) or do not have compatible cardinality restrictions, the resulting correspondence mapping is *underspecified.* That is, there may be many potential data transformations that can be generated from the correspondences. As future work, we want to further explore approaches for generating data-transformation programs from correspondence mappings, when the above assumptions do not hold.

5 Related Work

In this paper, we have presented a generic framework for enabling service composition in scientific workflows. The framework exploits three distinct specifications for services: structural types, semantic types, and registration mappings (which link structural and semantic types). We believe that separating these three specifications is a practical solution for enabling the reuse of legacy services. Also, under certain assumptions, registration mappings can be used to automatically generate desired structural transformations, allowing scientists to easily connect heterogeneous, but semantically similar services to form new workflows.

Although our goal is to integrate workflow services, our framework attempts to solve a different problem than typical data-integration systems

[Ul197, PB03, LGM01, PS98, PAGM96]. In particular, we want to construct data transformations that can take valid data under one schema, and convert it to valid data under another schema. In data integration, the focus is on combining source data expressed in heterogenous local schemas into an integrated view under a global schema. Note that some work has explored the use of ontologies as global schemas for data integration [SSR94, LVL03].

A number of languages have been defined for explicitly expressing data transformations between heterogeneous schemas [KLK91, CDSS98, DK97]. However, in scientific workflows, we believe it is important to compute data transformations whenever possible, as opposed to requiring scientists to express transformations manually, each time a connection is required. We note that a particular service may be used in a number of workflows, which would require a distinct data transformation mapping for each desired connection.

Instead, we use registration mappings from structural to semantic types (where a semantic type is similar to a global schema) to generate correspondence mappings. A number of recent approaches have explored the use of correspondences in data integration and transformation [PS98, PB03, PVM$^+$02]. In Clio [PVM$^+$02], a user specifies a set of correspondences between items in two structurally heterogeneous schemas, and using the constraints of the schemas, the system attempts to fill in, or generate, a complete mapping. Similarly, Pottinger and Bernstein [PB03] use simple correspondences to merge structurally heterogeneous schemas for data integration.

6 Future Work

We plan to further develop and extend the ontology-driven framework described here within the context of SEEK. Our next step is to extend Ptolemy with the XML-based framework described in Section 4. As part of this work, we also want to explore the relationships between correspondence mappings and the generation of data-transformation programs. In particular, we are interested in developing techniques to help generate data transformations for underspecified correspondence mappings and to extend the approach to support additional semantic conversions, for example, to automatically perform unit conversion. We also believe a more complex structural-type query language may be required for complex structural conversions, and intend to extend the simple XPath-based approach presented here as needed. Finally, we plan to investigate how the use of different computation models and workflow scheduling algorithms can influence, and help generate, data transformations.

References

[ABH$^+$02] Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry R. Payne, and Katia Sycara. DAML-S: Web service description for the semantic web. In *The First International Semantic Web Conference (ISWC),* June 2002.

[BCD$^+$02] Shuvra S. Bhattacharyya, Elaine Cheong, John Davis II, Mudit Goel, Christopher Hylands, Bart Kienhuis, Edward A. Lee, Jie Liu, Xiaojun Liu, Lukito Muliadi, Steve Neuendorffer, John Reekie, Neil Smyth, Jeff Tsay, Brian Vogel, Winthrop Williams, Yuhong Xiong, and Haiyang Zheng. Heterogeneous concurrent modeling and design in java. Technical Report Memorandum UCB/ERL M02/23, EECS, University of California, Berkeley, August 2002.

[BCF$^+$03] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniel Florescu, Jonathon Robie, and Jérôme Siméon, editors. *XQuery 1.0: An XML Query Language.* W3C Working Draft. World Wide Web Consortium (W3C), November 2003. http://www.w3.org/TR/2003/WD-xquery-20031112/.

[BG03] Dan Brickley and R.V. Guha, editors. *RDF Vocabulary Description Language 1.0: RDF Schema.* W3C Working Draft. World Wide Web Consortium (W3C), February 2003. http://www.w3.org/TR/2003/WD-rdf-schema-20030123/.

[BHT96] Michael Begon, John L. Harper, and Colin R. Townsend. *Ecology: Individuals, Populations, and Communities.* Blackwell Science, 1996.

[BM01] Paul V. Biron and Ashok Malhotra, editors. *XML Schema Part 2: Datatypes.* W3C Recommendation. World Wide Web Consortium (W3C), May 2001. http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[BN03] Franz Baader and Werner Nutt. Basic description logics. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, editors. *Web Services Description Language (WSDL) 1.1.* W3C Note. World Wide Web Consortium (W3C), March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[CD99] James Clark and Steve DeRose, editors. *XML Path Language Version 1.0.* W3C Recommendation. World Wide Web Consortium (W3C), November 1999. http://www.w3.org/TR/1999/REC-xpath-19991116.

[CDSS98] Sophie Cluet, Claude Delobel, Jérôme Siméon, and Katarzyna Smaga. Your mediators need data conversion! In *Proceedings of the SIGMOD International Conference on Management of Data,* pages 177–188. ACM Press, 1998.

[DK97] Susan B. Davidson and Anthony Kosky. WOL: A language for database transformations and constraints. In *Proceedings of the 13th International Conference on Data Engineering (ICDE),* pages 55–65. IEEE Computer Society, 1997.

[KLK91] Ravi Krishnamurthy, Witold Litwin, and William Kent. Language features for interoperability of databases with schematic discrepancies. In *Proceedings of the SIGMOD International Conference on Management of Data,* pages 40–49. ACM Press, 1991.

[LGM01] Bertram Ludäscher, Armanath Gupta, and Maryann E. Martone. Model-based mediation with domain maps. In *Proceedings of the 17th International Conference on Data Engineering (ICDE),* pages 81–90. IEEE Computer Society, April 2001.

[LP95] Edward A. Lee and Thomas M. Parks. Dataflow process networks. *Proceedings of the IEEE,* 83(5):773–801, 1995.

[LVL03] Fereidoon Sadri Laks V.S. Lakshmanan. Interoperability on XML data. In *Proceedings of the International Semantic Web Conference (ISWC)*, volume 2870, pages 146–163. Lecture Notes in Computer Science, 2003.

[Mic03] William K. Michener. Building SEEK: the science environment for ecological knowledge. In *DataBits: An electronic newsletter for Information Managers*, 2003. Spring Issue.

[MS02] Gerome Miklau and Dan Suciu. Containment and equivalence for an XPath fragment. In *Proceedings of the 21st Symposium on Principles of Database Systems (PODS)*, pages 65–76. ACM Press, June 2002.

[MvH03] Deborah L. McGuinness and Frank van Harmelen, editors. *OWL Web Ontology Language Overview*. W3C Candidate Recommendation. World Wide Web Consortium (W3C), August 2003. http://www.w3.org/TR/2003/CR-owl-features-20030818/.

[PAGM96] Yannis Papakonstantinou, Serge Abiteboul, and Hector Garcia-Molina. Object fusion in mediator systems. In *Proceedings of 22nd International Conference on Very Large Data Bases (VLDB)*, pages 413–424. Morgan Kaufmann, September 1996.

[PB03] Rachel Pottinger and Philip A. Bernstein. Merging models based on given correspondences. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB)*, pages 826–837. Morgan Kaufmann, September 2003.

[PS98] Christine Parent and Stefano Spaccapietra. Issues and approaches of database integration. *Communications of the ACM*, 41(5):166–178, May 1998.

[PVM$^+$02] Lucian Popa, Yannis Velegrakis, Renée J. Miller, Maricio Hernández, and Ronald Fagin. Translating Web data. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB)*, 2002.

[SSR94] Edward Sciore, Michael Siegel, and Arnon Rosenthal. Using semantic values to falilitate interoperability among heterogeneous information systems. *ACM Transactions on Database Systems*, 19(2):254–290, 1994.

[TBMM01] Henry S. Thompson, David Beech, Murray Maloney, and Noah Mendelsohn, editors. *XML Schema Part 1: Structures*. W3C Recommendation. World Wide Web Consortium (W3C), May 2001. http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *Proceedings of the International Conference on Database Theory (ICDT)*, volume 1186, pages 19–40. Lecture Notes in Computer Science, 1997.

PROVA: Rule-Based Java-Scripting for a Bioinformatics Semantic Web

Alexander Kozlenkov[1] and Michael Schroeder[2]

[1] City University, London, UK
[2] Biotec/Dept. of Computing, TU Dresden, Germany
ms@mpi-cbg.de

Abstract. Transparent information integration across distributed and heterogeneous data sources and computational tools is a prime concern for bioinformatics. Recently, there have been proposals for a semantic web addressing these requirements. A promising approach for such a semantic web are the integration of rules to specify and implement workflows and object-orientation to cater for computational aspects.
We present PROVA, a Java-based rule-engine, which realises this integration. It enables one to separate a declarative description of information workflows from any implementation details and thus easily create and maintain code. We show how PROVA is used to compose an information and computation workflow involving

- rules for specifying the workflow,
- rules for reasoning over the data,
- rules for accessing flat files, databases, and other services, and
- rules involving heavy-duty computations.

The resulting code is very compact and re-usable.
We give a detailed account of PROVA and document its use with a example of a system, PSIMAP, which derives domain-domain interactions from multidomain structures in the PDB using the SCOP domain and superfamily definitions. PSIMAP is a typical bioinformatics application in that it integrates disparate information resources in different formats (flat files (PDB) and database (SCOP)) requiring additional computations.
PROVA is available at comas.soi.city.ac.uk/prova

Keywords: Rules, Reasoning, Declarative and Object oriented Programming, Workflows.

1 Background

Systems integration is vital for progress in bioinformatics. Currently, users typically use systems via the web limiting their possibilities. While there are systems, which integrate a number of databases and tools, such as SRS, Expasy, NCBI, etc. these systems are inflexible and it is not possible to freely customize information workflows. There are some efforts to tightly integrate data and computational resources such as Edit2Trembl [MLFA99, MSA01, MKA00], which integrates annotations of tools for transmembrane prediction, GeneWeaver [BLJJ00], which integrates genome annotations, MyGrid

[SRG03], which provides personalised access to bioinformatics resources focusing on service management and discovery as part of the workflows.

However, many approaches are hard-wired and cannot adapt to changes in the underlying information resources. They also rely on a hand-crafted mapping of schemata of the different information resources. To this end, the semantic web promises a solution to information integration through the use of shared ontologies, which serve as common reference point for the information resources. Essentially, an ontology is a hierarchically structured taxomomy, that maps out a domain. In bioinformatics, GeneOntology [Con00] is a very prominent effort that defines a vocabulary for molecular biology covering processes, functions, and location. Besides ontologies, a semantic web will contain web services, which extend traditional web pages in that they allow access to remote computations and resulting data.

Semantic web that consists of data specified with a hierarchical common ontology and web services for computations can be programmed by marrying concepts from declarative and object-oriented programming. Rules, reasoning, and logic are needed for dealing with ontologies and for specifying workflows and relationships between the data. Object-orientation is needed for reflecting the hierarchy of the ontology and for implementing services and computations. Rule-based programming has another advantage. It transparently extends the expressiveness and power of relational databases that are often underlying bioinformatics databases. Rules serve as virtual table generators that derive knowledge from elementary facts.

Rule-based programming and object-orientation serve complementary purposes. Declarative programming style is very good for flexible data integration, inference, and for specifying if-then logic. Object-orientation is optimal for modeling complex systems that are relatively stable and well understood. Declarative programming is good for logic, while object-orientation is good for computation.

PROVA, the system presented in this paper, offers a clean way of integrating object-oriented Java with rule-based programming, paving a way for intelligent knowledge-intensive data integration useful in knowledge discovery for biomedical research.

2 Motivation

Before we formulate requirements for a more flexible and fruitful bioinformatics web and how our language PROVA meets these requirements, let us analyse the available data and their formats in detail. Although many bioinformatics data resources are kept in flat files or more recently in XML, their nature is relational. End users typically download these files, define database schemes and populate the databases with the data.

Example 1. Relational Databases
All major data source such as e.g. PDB, SCOP, GeneOntology, DIP, SWISSPROT, etc. can be stored as relational tables. Let us consider two examples. PDB [Tea03], the protein databank, stores atomic coordinates of structures determined by X-Ray christallography and nuclear magnetic resonance. Part of a PDB database will have a table with the following schema

PDB: PDB_ID, Atom_ID, Atom type, x, y, z

SCOP [MBHC95], the structural classification of proteins, is based on PDB and hierarchically groups domains of proteins according to their structural composition. A SCOP database contains a table

Scop: PDB_ID, superfamily, family, domain

PSIMAP [DBG+03, BDH+03] is a database of structural domain-domain interactions derived from PDB and Scop. Such interactions can be stored in a table:

PSIMAP: superfamily, superfamily

Once data is available in a relational database format rather than the original flat files, it can be queried using SQL, the structured querying language.

Example 2. SQL queries
SQL could answer queries such as the following:

- PDB: What are the atomic coordinates of the structure with PDB_ID 1 goj?
- Scop: How many domains does structure 1goj have?
- Scop: Does 1goj have a domain belonging to the P-loop superfamily?
- Interaction: What are the interaction partners of the P-loop?

However, sometimes SQL is not sufficiently expressive to answer queries.

Example 3. Expressiveness of SQL
SQL does not have the same computational expressiveness as a full programming language. In particular, it cannot express transitive closure. Hence, the queries

- Interaction: List all superfamilies the P-loop can directly or indirectly interact with

There is another problem. Most useful queries will involve more than one data source and thus there have to be joins across databases, which can be distributed.

Example 4. Joins across (distributed) databases

- SCOP/PDB: List all P-loop domains whose resolution is better than 1.5 Angstrom
- SCOP/PDB: For PDB entry 1 goj list all atomic coordinates separated by their Scop domains

Finally, many queries will have a computational component to them involving additional tools and computations that process the integrated data.

Example 5. Computations
SCOP/PDB: For a given multi-domain structure in PDB, check for all its pairs of Scop domains whether these domains have at least 5 residue pairs within 5 Angstrom and can hence be said to interact.

Let us summarise the above observations.

1. Relational databases: Most information sources used in bioinformatics can be specified as relational databases.
2. Simple queries: For many purposes, SQL-type querying is useful and sufficient.

3. Expressive queries: Some queries require more expresiveness than SQL
4. Remote access: Resources are often distributed, but accessible through interfaces such as HTML, database server, or web services.
5. Computation: As a rule, information integration also has computation-intensive components.

These observations can serve as requirements, which need to be met to achieve flexible and transparent systems integration. Additionally, a prime concern in systems integration must be the separation of the workflow and the details of the data and computation resources to be integrated.

Rules are a natural extension of relational databases to achieve more expressiveness.

Example 6. Rules as workflows
PSIMAP [DBG+03, BDH+03] considers each multi-domain protein in PDB and establishes an interaction between a pair of domains if they have at least 5 residue pairs within 5 Angstrom. An implementation of PSIMAP will have some complexity, as PDB and Scop need to be wrapped and accessed. Most imperative approaches will mix the workflow with imlementation details such as wrappers and the actual computation. However, using rules the workflow itself can be specified as a single simple rule:

```
PSIMAP(SF1,SF2) if
  PDB(PDB_ID),
  Scop(PDB_ID,SF1,F1,D1),
  Scop(PDB_ID,SF2,F2,D2),
  D1<>D2,
  interact(D1,D2)
```

Superfamilies `SF1` and `SF2` interact if there is a PDB entry `PDB_ID` for which there is a Scop domain `D1` of superfamily `SF1` and a Scop domain `D2` of superfamily `SF2`, where `D1` and `D2` are different, and the two domains interact.

This is the pure workflow and it does not yet specify how the PDB entry is obtained, where the Scop data comes from and how the computation is carried out. These are all implementation details that are defined by separate rules, which in turn may require access to a database or system calls.

Besides using rules to specify workflows, rules can be used to infer relationships from the data. As shown in example 3, we might wish to know all superfamilies a given superfamily such as the P-loop is directly or indirectly connected to.

Example 7. Inference and Rules
Two rules are needed to define if two superfamilies `X` and `Y` are connected, i.e. if they can interact directly or indirectly. First, `X` and `Y` are connected if they interact (directly). Second, `X` and `Y` are connected if `X` interacts directly with a third superfamily `Z`, which is connected to `Y`. In rule format, this looks as follows:

```
connected(X,Y) if interact(X,Y).
connected(X,Y) if interact(X,Z), connected(Z,Y).
```

3 PROVA

Our system PROVA, which is based itself on Mandarax [Die], addresses the above requirements by providing a rule-based Java scripting language. The use of rules allows one to declaratively specify the integration needs at a high-level without any implementation details. The transparent integration of Java caters for easy access and integration of database access, web services, and many other Java services. This way PROVA combines the advantages of rule-based programming and object-oriented programming in Java. This section presents rationales and design principles behind the PROVA language. We position this language as a platform for knowledge-intensive ontology-rich applications in biomedical research. We are aiming to satisfy the following design goals with the proposed PROVA language:

- Combine the benefits of declarative and object-oriented programming;
- Merge the syntaxes of Prolog, as rule-based language, and Java as object-oriented languages;
- Expose logic as rules;
- Access data sources via wrappers written in Java or command-line shells like Perl;
- Make all Java API from available packages directly accessible from rules;
- Run within the Java runtime environement;
- Be compatible with web- and agent-based software architectures;
- Provide functionality necessary for rapid application prototyping and low cost maintenance.

Consider the following PROVA code showing how knowledge can be inferred from the available facts.

Example 8. (Declarative programming)
Consider a table of interacting proteins. We wish to infer all interactions, direct or indirect. In PROVA, this can be specified as follows (: - is read as "if"):

```
% Facts (what we know)
interactDirect(a,b).
interactDirect(b,c).
interactDirect(c,d).

% Rules (how to derive new knowledge)
interact(X,Y):-interactDirect(X,Y).
interact(X,Z):-interactDirect(X,Y),interact(Y,Z).
```

The query `: - solve(interact(a,X)).`, which can be read as "which proteins `X` interact with protein `a`?", will return the three answers `x=b`,`X=c`,and `X=d`.

Thus, PROVA follows classical Prolog closely by declaratively specifying relationships with facts and rules. Now let us consider two examples, where access to Java methods is directly integrated into rules.

Example 9. (Object-oriented programming)
The code below represents a rule whose body consists of three Java method calls: the first to construct a String object, the second to append something to the string, and the third to print the string to the screen.

```
hello(Name):-
  S = java.lang.String("Hello "),
  S.append(Name),
  java.lang.System.out.println(S).
```

Java method calls can be used to wrap up computations. As an example, consider the PSIMAP application, in which domain-domain interactions are computed by checking whether 5 residue pairs are within 5 Angstrom. The exact PROVA code to carry out this computation will be dicussed later. However, as an example for method call, consider the line below in which the Java method `interacts` of object `DomainA` is invoked with parameter `DomainB`.

```
scop_dom2dom(...):-
  ...
  DomainA.interacts(DomainB).
```

Let us consider the PROVA syntax in detail. The first example above shows how an object is constructed. The code is very similar to Java itself, only the `new` keyword is missing. Once an object is available in PROVA, methods can be invoked on it in the same way as in Java including both instance (`DomainA.interacts`) and static calls (`java.lang.System.out.println`). The latter needs fully qualified class names. An instance method can fail if an exception is raised or if the unification of the optional returned object with the supplied pattern fails. There are two important rules about the passing and returning of PROVA lists from Java methods. Essentially, all PROVA lists are automatically converted to the Java ArrayList implementing the List interface when passed to a Java method requiring a Collection for the corresponding parameter. Conversely, when a Java Object array (`Object[]`) is returned from a Java method, it is automatically converted into a PROVA list.

The full PROVA syntax in EBNF is shown in appendix A.

3.1 Programming with PROVA

Let us consider two example rules taken from the implementation of PSIMAP, the protein structure interaction map introduced earlier.

Given the general approach of decomposing an application workflow into the logic and computational parts, we capture the knowledge-intensive part of the application in PROVA rules directly while using Java, SQL, or command-line utilities based services for accessing or generating data for which the highest level of performance is needed and the logical component of the query is minimal. Consider the code fragment below, which implements the following definition: "If there is a PDB entry and according to SCOP this entry has two domains and according to the computation `interacts` at least 5 residue pairs are within 5 Angstrom, then these two domains interact."

```
% Given the open database connection DB
% and a unique protein identifier in Protein
% Data Bank PDB_ID, test whether the provided
% domains with IDs PXA and PXB interact
% (have at least 5 atoms within 5 angstroms)
scop_dom2dom(DB,PDB_ID,PXA,PXB) :-
  access_data(pdb,PDB_ID,Protein),
  scop_dom_atoms(DB,Protein,PXA,DomainA),
  scop_dom_atoms(DB,Protein,PXB,DomainB),
  DomainA.interacts(DomainB).
```

The code shows an example of a data integration rule used for computing the interaction of protein domains. The scop_dom2dom-predicate in the head of the rule represents a virtual table (view) composed from computational wrappers and lower-level predicates. The body of the rules includes a generic cache-based access to PDB data via the predicate access_data, which is discussed below and which implements an active cache. Given a PDB entry, a pair of domains is returned by the predicate scop_dom_atoms and returned in the variables DomainA and DomainB, respectively. The predicate requires a variable DB, which contains a database connection from where the data is read. The use of databases and SQL is discussed below. Finally, an instance method interacts returning boolean either succeeds or fails depending on whether the two domains interact or not.

The example shows how external Java based data wrappers fit into the rule-based data integration. Java variables can be constructed and returned from predicates associated with Java calls while boolean methods can be used to test conditions.

Now let us consider how the caching of PDB is implemented. The code below shows how logic representation helps dealing with the complexities of defining the caching mechanism for retrieving any type of data. We represent the data items by the data type (variable Type) and a unique ID of a particular data item (variable ID). The data is stored in the cache represented by variable CacheData.

```
% Top-level rule for accessing Data
% given data Type and particular ID.
access_data(Type,ID,Data) :-
  % Retrieve or create the cache for Type
  cache(Type,CacheData),
  access_data(Type,ID,Data,CacheData).

% Two alternative rules for either retrieving data
% from the cache or accessing the data from its
% original location and caching it.
access_data(Type,ID,Data,CacheData) :-
  % Attempt to retrieve the data
  Data=CacheData.get(ID),
  % Success, Data (whatever object it is) is returned
  !.
```

```
access_data(Type,ID,Data,CacheData) :-
  % Retrieve the data from its location and update
  % the cache
  retrieve_data_general(Type,ID,Data),
  update_cache(Type,ID,Data,CacheData).
```

The code above demonstrates a very high level of abstraction and flexibility offered by rule-based systems. In particular, the cache variable is untyped which means that if implementation for storing a cache changes the code will remain valid. Also, both the ID and the Data itself are also untyped which allows one to use for instance complex IDs represented as lists for unique IDs of the data items. The last two rules show how IF-THEN logic is realised as alternative rules. If `Data=CacheData.get(ID)` succeeds, a cut operator ('!') prevents further backtracking to another rule. Otherwise, the `retrieve_data_general` predicate is used for fetching the data from its original location. If there were several mirrors available for the original data, they are automatically explored until the working mirror is identified.

The procedural code representing this type of logic would have been much more cumbersome to write and maintain. Enhanced level of generality and flexibility is exactly what is required for real-world data integration and manipulation projects involving access to multiple rapidly changing data sources.

3.2 Typing in PROVA

Because of the desired tight integration of PROVA rules with Java code and extended use of types and classes in Java, we have decided to include a type system in PROVA. This language feature is not commonly found in rule-based languages such as Prolog. The rationale behind the introduction of the type system was to offer the user the option to restrict the applicability of rules and to control the level of generality in queries. The notation for typed variables in PROVA normally involves prefixing a variable name with a fully qualified name of the class to which the variable should belong. All classes in the java.lang package can be used without their full prefix specifying their package. Consider a `member` predicate returning the members of a list. The query `member(X,[1,Double.D,"3"]))`, will return `X=1`, `X=java.lang.Double.D`, and `X=3`. The query variable `X` could be further qualified as an integer. Then the query `member(Integer.X,[1,Double.D,"3"]))` will return only `java.lang.Integer.X = 1`, as the other assignments fail as they are double and string, respectively.

In the first query, an untyped variable `X` is used to query for list members. The first variable in the target list is an Integer constant, while the second one is a Double variable, and the last one is a string constant. Three solutions of this query demonstrate this. The second query asks specifically for an Integer list members specifying an Integer variable as the first argument. As a result, only the first element, a constant 1, is returned.

When more complex Java classes are used in PROVA, the following rules apply to variable-variable unification. If the query and target variable are not assignable to each other, the unification fails. Otherwise, the unification succeeds. If the query variable belongs to a subclass of the class of the target variable, the query variable assumes the type of the target variable. If the query variable belongs to a class that is a superclass of the class of the target variable or is of the same class as the target variable, the query variable retains its class.

3.3 PROVA SQL Integration

PROVA SQL integration has a crucial role in providing an efficient and flexible mechanism for data and ontology integration. PROVA offers a seamless integration of predicates with most common SQL queries and updates. The language goes beyond providing embedded SQL calls and attempts to achieve a more flexible and natural integration of queries with PROVA predicates.

```
:-eval(consult("utils.prova")).
:-eval(test()).

location(database,"jdbc string", "database_name",
                                "username","password").
test():-dbopen("database_name",DB).
```

Opening a database only requires calling the dbopen predicate and providing the database name. The rules for the dbopen predicate together with accompanying rules for caching and mirroring are provided with the supplied external module utils.prova that needs to be included (consulted) at the beginning of a user file. Opening database requires one or more location records to be present in the fact base. The location records help organising the information about possible alternative locations of data sources. The data sources are organised according to their type, name, and any required keys for a particular dataset to be retrieved. Caching for datasets is automatically provided.

In the case of opening a database, the data source type is database, the data source name is the database name provided as the second argument to location, a JDBC connection string is provided as the third argument, and optional username and password can be provided as strings. If more than one record for a particular database is provided, the system attempts to establish connection with the each one in turn until a successful connection is established or all locations are exhausted and opening the database fails. This mirroring technique is especially useful for web-intensive applications where configuration flexibility is needed. For example, if an application is developed on one computer and then deployed on the web server, the rule system above can be used to achieve complete code independence. The example below illustrates this situation.

```
:-eval(consult("utils.prova")).

location(database,scop,"jdbc:mysql://comas.soi.city.ac.uk",
        "u","p").
location(database,scop,"jdbc:mysql://localhost","u","p").
dbopen(scop,DB)
```

By default open database connections are cached. If the users wish to override the default on how many database connections are cached they can override the corresponding fact for the predicate `cache_capacity`.

The main format for PROVA predicates dynamically mapped to SQL Select statements is shown below:

```
sql_select(DB,Table,[N1,V1],...,[Nk,Vk],
      [where,Where],[having,Having],[options,Options])
```

The built-in `sql_select` predicate non-deterministically enumerates over all possible records in the result set corresponding to the query. The predicate fails if the result set is empty or an exception occurs. It accepts a variable number of parameters of which only the first two are required. DB corresponds to an open database connection and Table is the name of the table to be queried. Note that the table name can be a variable that only becomes instantiated during the execution of the code. Not only the table name can be determined dynamically, but also all the remaining parameters can be either variables or constants or even the whole list of parameters can be dynamically constructed.

The most important part of the syntax of `sql_select` is 0 or more field name-value pairs `[N1,V1],...,[Nk, Vk]`. `N1,...,Nk` correspond to field names included in the query. As opposed to ordinary SQL Select statements, this list of fields includes both the fields to be returned from the query and those that can be supplied in the SQL Where clause. Whether a particular field `Ni` will be returned or used as a constraint depends on the values `Vi` corresponding to these field. If `Vi` is a constant at the time of the invocation, it becomes a constraint in the automatically constructed Where clause. Otherwise, `Vi` is an uninstantiated (free) variable and will be returned by the query in each record in the result set. In addition to simple field names, `N1,...,Nk` can be strings containing special SQL modifiers such as Distinct (for example, "distinct name") or group functions such as Count (for example, "count(px)").

The remaining parameters are entirely optional. In the pair `[where, Where]`, "where" is a reserved word and `Where` is a variable or constant containing an explicit SQL Where clause. This syntax is useful in situations requiring the use of such constraints as `Like` or `Rlike`, for example, `[where, "pdb_id like '%%gs'"]`. The pair `[having, Having]` allows specifying a post processing filter on the results returned by the query, for example, `[having,"count(px)>1"]`. A large variety of other modifiers for the query can be included with the `[options, Optioons ]` pair, for example, `[options,"order by count(px) desc limit 10"]`. A number of examples of SQL Select mapped predicates working with SCOP are shown below.

```
sql_select(DB,cla,[pdb_id,"1alx"],[px,Domain])
sql_select(DB,cla,[pdb_id,PDB_ID],[count(px),2])
sql_select(DB,cla,[pdb_id,PDB_ID],[count(px),Count])
sql_select DB,cla,[pdb_id,PDB_ID],[count(px),Count],
   [where,"pdb_id like '%%gs'"]
sql_select(DB,cla,["distinct pdb_id",PDB_ID],
   [options,"limit 10"])
```

Although only queries corresponding to a single table are supported at this moment, queries with joined tables can be constructed by combining several single table queries. A future optimisation will provide an automatic construction of joins from `sql_select` predicates that have one or more common variables. The example below shows how two `sql_select` calls can be used to compute an inner join for table `cla` finding two different domains `PXA` and `PXB` belonging to the same PDB file.

```
sql_select(DB,cla,[px,PXA],[pdb_id,PDB_ID]),
sql_select(DB,cla,[px,PXB],[pdb_id,PDB_ID]),
PXA<PXB
```

PROVA provides a built-in predicate `sql_insert` providing a flexible mapping to SQL Insert statements.

```
sql_insert(DB,Table,[N1,...,Nk],[V1,...,Vk])
```

The `sql_insert` predicate is structured differently from `sql_select` in that it accepts the field names as a separate sublist in the third argument and the fourth argument contains a sublist with values corresponding to these fields. The example below shows a complete rule that parses a text-based database file with descriptions of protein domains from the SCOP database.

```
db_import(DB,scop,des,Line) :-
    tokenize_list(Line,"\t",[T|Ts]),
    sql_insert(DB,des,[id,type,sccs,sid,description],
               [T|Ts]).
```

The predicate `db_import` receives an open database connection `DB`, the name `scop` of the database to be imported, the name of the table to insert records into, and a `Line` from the text file. The built-in predicate `tokenize_list` builds `Line` tokens separated by tab characters and outputs them in the list `[T|Ts]`. Finally, `sql_insert` inserts a new record into the table `des` with the specified fields and the values copied from the list of tokens.

4 Comparison and Conclusion

Rules play an important role in bioinformatics. They come in many guises such as database views, logic programs, constraints, active rules, description logics, ontologies, etc. They have been used in many different contexts such as constraints for structure prediction [BWBB99, KB02, PKWM00], model checking of biochemical networks [CF03], logic programming for consistent annotation of proteins [MLFA99, MSA01, MKA00], ontologies for transparent access to bioinformatics databases [LJ03, LE03], ontologies for health applications [GE01, GES01], and integration of an ontology with gene expression data [Bad03, BT03]. All but the first two applications involve systems integration and therefore showcase the usefulness of rules, logic and ontologies for this task.

In this paper, we have argued that rules done are not sufficient to easily facilitate systems integration. We argued that a rule-based approach also needs to cater for computation, database access, communication, web services, etc. All of these requirements can be met by integrating a rule-based approach with an object-oriented approach such as Java. The challenge is to make this integration transparent and get the best out of two worlds.

There are a number of approaches that integrate object-orientation and deduction such as Rock&Roll [FWPM94], ConceptBase [JGJ+95], and Flora [YK00]. These approaches aim for a conceptually coherent integration of the two different programming paradigmns. In contrast to this, PROVA aims to provide a rule-engine for Java, which caters for rule-based processing in Java. At first glance, this objective seems to be addressed also by catering for native method calls in Prolog engines. as exemplified by systems such as JIProlog. There is however a difference to PROVA: PROVA emphasises flexible integration as opposed to achieving simple interfacing of Java with Prolog. In the following, we provide some comments for features listed in the table. This different perspective leads to a host of features absent from JIProlog, but provided by PROVA such as native syntax Java call, Java type system within the rules, access to static java variables from the rules, automatic conversion of returned Java object lists into Prolog lists, and interpretation instead of compilation. Furthermore, PROVA provides features specifically useful for systems integration, namely variable arity and argument tails, flexible database access, predicate names as variables, message-passing and reaction rules for implementing active behaviour.

All in all, PROVA is a first step towards realising a semantic web for bioinformatics by declaratively and transparently integrating data and rules including database access and computations captured in procedural (Java) code. PROVA facilitates the separation of declarative workflows from implementation details and thus leads to more compact and maintainable code. We have shown how all of the above features are used to implement the PSIMAP system, which computes domain-domain interactions from SCOP and PDB.

References

[Bad03] Liviu Badea. Functional discrimination of gene expression patterns in terms of the Gene Ontology. In *Proceedings of the Pacific Symposium on Biocomputing - PSB03,* 2003.

[BDH+03] Dan Bolser, Panos Dafas, Richard Harrington, Jong Park, and Michael Schroeder. Visualisation and graph-theoretic analysis of the large-scale protein structural interactome network psimap. *BMC Bioinformatics,* 4(45), 2003.

[BLJJ00] K. Bryson, Michael Luck, Mike Joy, and D. T. Jones. Applying agents to bioinformatics in geneweaver. In *Cooperative Information Agents,* pages 60–71, 2000.

[BT03] Liviu Badea and Doina Tilivea. Integrating biological process modelling with gene expression data and ontologies for functional genomics (position paper). In *Proceedings of the International Workshop on Computational Methods in Systems Biology,* University of Trento, 2003. Springer-Verlag.

[BWBB99] Rolf Backofen, Sebastian Will, and Erich Bornberg-Bauer. Application of Constraint Programming Techniques for Structure Prediction of Lattice Proteins with Extended Alphabets. *Journal of Bioinformatics,* 15(3):234–242, 1999.

[CF03] Nathalie Chabrier and François Fages. Symbolic model checking of biochemical networks. In *Proceedings of the First International Workshop on Computational Methods in Systems Biology CMSB'03,* LNCS, Riverto, Italy, March 2003. Springer-Verlag.

[Con00] The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nat Genet,* 25:25–29, 2000.

[DBG+03] Panos Dafas, Dan Bolser, Jacek Gomoluch, Jong Park, and Michael Schroeder. Fast and efficient computation of domain-domain interactions from known protein structures in the PDB. In H.W. Frisch, D. Frishman, V. Heun, and S. Kramer, editors, *Proceedings of German Conference on Bioinformatics,* pages 27–32, 2003.

[Die] J. Dietrich. *Mandarax.* http://www.mandarax.org.

[FWPM94] A.A.A. Fernandes, M.H. Williams, N. Paton, and M.L. Maria. Object-Oriented Database Programming Languages Founded on an Axiomatic Theory of Objects. In *Workshop on Logical Foundations of Object-oriented Programming,* 1994.

[GE01] Rolf Grütter and Claus Eikemeier. Development of a Simple Ontology Definition Language (SOntoDL) and its Application to a Medical Information Service on the World Wide Web. In *Proceedings of the First Semantic Web Working Symposium (SWWS '01),* pages 587–597, Stanford University, California, July/August 2001.

[GES01] Rolf Grütter, Claus Eikemeier, and Johann Steurer. Towards a Simple Ontology Definition Language (SontoDL) for a Semantic Web of Evidence-Based Medical Information. In S. Quaglini, P. Barahona, and S. Andreassen, editors, *Artificial Intelligence in Medicine. 8th Conference on Artificial Intelligence in Medicine in Europe, AIME2001,* Cascais, Portugal, July 2001. Springer-Verlag.

[JGJ+95] M. Jarke, R. Gallersdörfer, M.A. Jeusfeld, m. Staudt, and Stefan Eberer. CConceptBase - a deductive object base for meta data management. *J. of Intelligent Information Systems,* February 1995.

[KB02] L. Krippahl and P. Barahona. PSICO: Solving Protein Structures with Constraint Programming and Optimisation. *Constraints,* 7(3/4):317–331, July/October 2002.

[LE03] P. Lambrix and A. Edberg. Evaluation of ontology merging tools in bioinformatics. In *Proceedings of the Pacific Symposium on Biocomputing - PSB03,* pages 589–600, 2003.

[LJ03] P. Lambrix and V. Jakoniene. Towards transparent access to multiple biological databanks. In *Proceedings of the First Asia-Pacific Bioinformatics Conference,* pages 53–60, Adelaide, Australia, 2003.

[MBHC95] A.G. Murzin, S.E. Brenner, T. Hubbard, and C. Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.,* 247:536–540, 1995.

[MKA00] S. Möller, E. V. Kriventseva, and R. Apweiler. A collection of well characterised integral membrane proteins. *Bioinformatics,* 16(12):1159–1160, 2000.

[MLFA99] S. Möller, U. Leser, W. Fleischmann, and R. Apweiler. EDITtoTrEMBL: a distributed approach to high-quality automated protein sequence annotation. *Bioinformatics,* 15(3):219–227, 1999.

[MSA01] S. Möller, M. Schroeder, and R. Apweiler. Conflict-resolution for the automated annotation of transmembrane proteins. *Comput. Chem.,* 26(1):41–46, 2001.

[PKWM00] P.N. Palma, L. Krippahl, J.E. Wampler, and J.J.G. Moura. BiGGER: A new (soft) docking algorithm for predicting protein interactions. *Proteins: Structure, Function, and Genetics,* 39:372–384, 2000.

[SRG03] Robert D. Stevens, Alan J. Robinson, and Carole A. Goble. mygrid: personalised bioinformatics on the information grid. In *Eleventh International Conference on Intelligent Systems for Molecular Biology,* volume 19, 2003.

[Tea03] The PDB Team. The protein data bank. In *Structural Bioinformatics,* pages 181–198. Wiley, 2003.

[YK00] G. Yang and M. Kifer. FLORA: Implementing an efficient DOOD system using a tabling logic engine. In *LNAi 1861.* Springer, 2000.

A Syntax of PROVA

```
                prova ::= statements, end of file;
           statements ::= statement, statements;
            statement ::= (fact — rule — query), end of statement;
                 fact ::= relation;
                 rule ::= relation, ”:-”, atoms;
                query ::= (”eval” — ”solve”), ”(”, relation, ”)”;
                atoms ::= atom, ”,”, atoms;
                 atom ::= relation — arithmetic relation — java call — cut;
             relation ::= predicate symbol, ”(”, terms, ”—”, argument tail, ”)”;
        argument tail ::= variable;
     predicate symbol ::= lowercase word — uppercase word;
            java call ::= functional java call — predicate java call — constructor java call;
 functional java call ::= left term, ”=”, predicate java call;
  predicate java call ::= static java call — instance java call;
     static java call ::= qualified java class,”.”,method name,”(”, terms,”)”;
   instance java call ::= variable, ”.”, method name, ”(”, terms, ”)”;
constructor java call ::= left term,”=”,qualified java class,”(”,terms,”)”;
                terms ::= term, ”,”, terms;
                 term ::= left term — (func, ”(”, terms, ”)”);
            left term ::= variable — constant — prova list;
                 func ::= variable — constant;
             variable ::= uppercase word — typed variable;
             constant ::= lowercase word — (””, string, ””) ;
       typed variable ::= qualified java class, uppercase word;
           prova list ::= ”[]” — (”[”, head, ”—”, tail, ”]”);
  arithmetic relation ::= left term, binary operator, term;
      binary operator ::= ”=” — ”!=” — ”¿” — ”¡” — ”¿=” — ”¡=”;
                 head ::= term;
                 tail ::= variable;
       uppercase word ::= [”A”-”Z”,”_”], lowercase word;
       lowercase word ::= [”a”-”z”], word;
                 word ::= [”a”-”Z”,0-9]+;
                  cut ::= ”!”;
     end of statement ::= ”.”;
```

Process Based Data Logistics: a Solution for Clinical Integration Problems*

Stefan Jablonski, Rainer Lay, Christian Meiler, Sascha Müller

Lehrstuhl für Datenbanksysteme - Institut für Informatik
Universität Erlangen-Nürnberg
Martensstraße 3
D-91058 Erlangen
{Stefan.Jablonski, Rainer.Lay, Christian.Meiler, Sascha.Mueller}
@informatik.uni-erlangen.de

Abstract. Integration is a big issue in healthcare environments. One aspect of integration is data logistics that supplies physicians with relevant patient data along treatment processes. The two main tasks facilitated by data logistics are data transportation and data transformation. To enable data transport, workflow management concepts are adopted for indirect process support to supply data. To enable data transformation, formats, ontologies and terminologies are considered in an XML based transformation approach. A case study regarding self tonometry illustrates this approach.

1 Introduction

In modern clinical environments many information systems are used simultaneously, for example laboratory systems, clinical information systems or patient information systems. In order to support clinical work, the systems have to communicate with each other, i.e. they must be integrated. As in many other application domains, communication in hospital and clinical supply networks has to be optimized in order to increase data accessibility and quality and to reduce cost and waiting time. Often specialized middleware, called connection server or integration engine, is used to establish the necessary communication relationships in a standardized format, e.g. Health Level 7 (HL7) [HaMo93]. Connection servers achieve this by translating between system specific protocols and by transporting data between producers and receivers. Nevertheless there are problems with communication in clinical environments. A broad variety of systems and applications – often showing overlapping functionality and data – makes this situation complex and complicated. One of the most important issues is to cope with the complexity of communication since often hundreds of communication scenarios have to be supported. Especially, when communication scenarios have to be changed, problems arise since changes often produce unexpected side effects which lead to failures.

* This project is funded by the Deutsche Forschungs Gemeinschaft (DFG), SFB539: Glaukome einschließlich Pseudoexfoliationssyndrom (PEX)

Connection servers and other approaches like the adaptive replication manager [NiHa02] or implementations of the publish/subscribe paradigm [EFGK03] provide solutions to enable and execute communication. But all these approaches show a deficit on the conceptual level. They are more or less implementation and not model driven, which limits the possibility of communicating details of the configuration between domain and implementation experts. In all these approaches a comprehensive model covering all communication scenarios is missing totally. Our method of process based data logistics uses well documented process models to describe the application scenarios. This also supports the idea of evidence based patient treatment according to medical guidelines and clinical pathways [BKMJ04].

In this paper we present an approach to make use of process based data logistics. In Section 2 we introduce basic technologies our approach relies upon: data transformation and workflow management. In Section 3 we present our approach before the concepts developed in this paper are illustrated in a case study and a prototypical implementation in Section 4.

2 Basic Technologies and Approaches

In this section we introduce the basic technologies and concepts that are needed on our way to process based data logistics. Firstly, we have to consider data transformation; it cannot be done without concerning syntax and semantics of data. In medical environments format, ontologies and terminology must be analyzed. Secondly, data transport must be facilitated. Therefore we apply workflow management concepts, however, not in the usual way.

2.1 Formats, Ontologies, and Terminologies

In a typical medical environment several data formats are used. HL7 is a wide spread but by far not ubiquitous standard that helps to solve the integration problems in healthcare. Data formats do not only differ on the syntactical level (How is the data encoded?), but also on an ontological level (How are the concepts described?) and terminological level (What terms are used?). These aspects [BKMJ04] are important when talking about integration in the sense that a target system is able to understand data sent by a source system. Let us have a closer look at these three aspects.

Different systems use different encoding rules to represent data. A lot of up-to-date standards are XML based, but older standards and a lot of systems in healthcare settings use proprietary data formats instead of XML. In general only a syntactical transformation is needed to transform the encoding of one data format into another.

In real word applications it is not sufficient to consider only the encoding on syntactical level. Systems use different concepts to describe an application area. For example, the way a medical finding is structured and described differs from system to system. These phenomena can also be found when considering database schemata. An application area can be modeled and mapped to tables in different ways. Thus concepts or ontologies used to express real world phenomena in information systems must be considered when integrating data from different systems.

But a syntactical and ontological integration on a conceptual level is not enough. Besides integration on schema level, integration on instance level is also necessary. On this level common terminologies are needed to support the users with a common language at run time, e.g. when preparing a finding report. This common language is necessary, to get a common understanding of domain specific terms used by medical information systems (e.g. findings and their meanings). Examples for such terminologies are ICD (International Classification of Diseases) and ICPM (International Classification of Procedures in Medicine).

Heterogeneity on these three levels can be reduced by using common standards. HL7 is such a standard and a framework for clinical communication that defines standardized formats and protocols used by connection servers. The focus of HL7 is to enable communication between information systems in hospitals. It consists of a Reference Information Model (RIM) that describes the entities and their relationships in a hospital, a vocabulary that holds a common definition of terms in different domains and message definitions that formalize data exchange [DAB+01]. Based on the RIM, HL7 defines the formats and semantics of messages that should be exchanged between different HL7 participants in order to submit information. The underlying intention of HL7 is to define a common understanding of the terms used by the participating information systems in order to improve interaction quality.

But because such standards are not used strictly in all systems, different ontologies must be considered and so far their integration is necessary. To summarize it shortly: integration in healthcare environments must be considered on syntactical, ontological and terminological level.

2.2 Workflow Management

In order to deploy a workflow management system (WfMS) in a hospital at least two major steps have to be performed: First a workflow model describing all aspects of the clinical process has to be designed and validated. Second, each participant of the workflow must be equipped with a work list that contains all tasks a WfMS is generating for him [JaBu96]. Participants of workflows then use these work lists in order to contribute to the execution of workflows instances that are derived from the workflow model.

There are a couple of advantages when workflow management systems are used in clinical environments. First, a WfMS can enforce the execution of predefined processes. Thus, the WfMS coordinates the involved information systems and human participants. Especially, the data transport between these information systems is organized by the WfMS. Therefore, a WfMS can be used in clinical environments in order to enact communication between clinical information systems. For instance, a WfMS can be used to exchange data between the three clinical information systems depicted in Fig. 1.

Second, a WfMS also enforces that necessary input information is available before a step can be performed. On the one hand, this is very advantageous in clinical processes. When – for instance – a physician has to execute a certain workflow step, it is guaranteed that the information he needs to perform his task is provided. He does not have to search for required data manually any more. However, this characteristic of a WfMS might also be disadvantageous in some situations. Let us assume that a

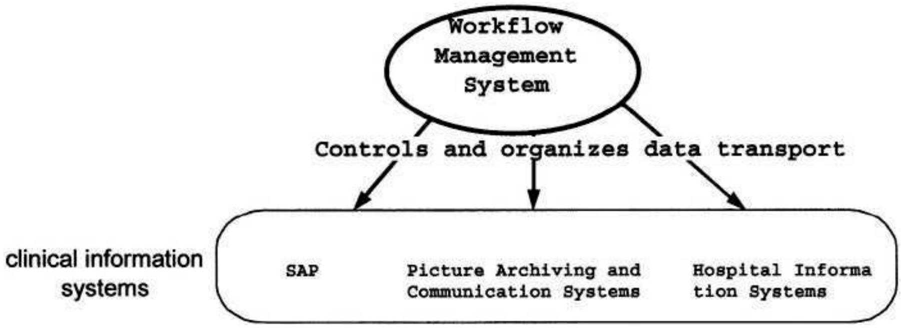

Workflow Management System
Controls and organizes data transport
clinical information systems
SAP
Picture Archiving and Communication Systems
Hospital Informa tion Systems

Fig. 1: WfMS control clinical information systems

surgery process is modeled in a way that it can only be started (within the WfMS), when the patient's insurance is specified. Then look at an emergency situation, where the patient is unconscious. The WfMS might not be able to execute the process, i.e. the surgery will take place – it cannot be rejected or postponed just due to the missing data – without support of the WfMS. This example shows that WfMS are not suitable to support many clinical processes as shown in the scenario above.

The third advantage of WfMS is the documentation gathered during the execution of a workflow. Since WfMS assign tasks to user groups or roles, it is later possible to check out who performed what step in the process.

Due to this need of having to specify a workflow model, workflow management delivers an abstract documentation of clinical communication. Other data communication approaches in healthcare, e.g. most connection servers [LaPH99] only focus on the implementation of the communication. These approaches are lacking an abstract communication model. So it is very hard to get a comprehensive overview of the communication needs of an application on the conceptual level.

The strict way of executing workflows (i.e. only decisions explicitly modeled are possible) works out well in environments like insurances or banks, which are characterized by a well known and limited number of applications and processes. Furthermore, the strictness of workflows also better supports well structured processes than processes that are very flexible (i.e. they have to be changed very often and show many alternative execution paths) like in clinical environments. There have been several approaches to ease this inherent restriction of the workflow approach. For example the concept of ad hoc workflows [HeSS96] or dynamic changes of workflow models [ReDa97] allow bypassing certain steps of the workflow in special cases. Unfortunately these extensions of the standard workflow model increase the complexity of the resulting workflow models even more.

Considering this lack of flexibility, it turns out that workflow management is not really suitable for the clinical application domain when clinical processes are directly enacted as workflows. Therefore, we considered another approach of using workflow management in clinical environments. We do not directly execute clinical processes as they are specified in workflow models, but derive data logistics requirements from these workflow models. The generated data logistics processes are then executed by a WfMS. This approach eliminates the discussed drawbacks of workflow management but still sustains its advantages as we will present in Section 3.

3 Data Logistics Approach

In Section 3.1 we introduce our principal approach of data logistics. The two issues resulting from data logistics, data transformation and data transport, are then discussed in Section 3.2 and Section 3.3. In Section 3.4 we shortly show how a data logistics workflow is built.

3.1 Principal Approach

The approach presented in the following takes into account the findings of Section 2 which are motivated by the observations of Section 1:

- Due to its complexity clinical applications need a comprehensive model in order to cope with its complexity. From such a model an execution system is derived. Process models are most suitable for modeling clinical applications since they can cope with this enormous complexity and ideally reflect clinical pathways.
- It would be self-evident to execute the processes describing clinical application through WfMS. However, due to its strictness WfMS are not adequate to execute the generated process models directly.

The problem we are facing in clinical environments can therefore be formulated like this: How can we support clinical applications that are described by process models without directly executing these processes by a WfMS? As an answer to this question, we propose to apply a new technique which is called data logistics. How does this approach work?

The approach assumes that clinical applications are described through process models. Among other things, these processes contain the communication requirements between clinical applications. The approach now distills these data communication requirements between clinical applications out of the process models. We will see that these communication requirements can also be represented and enacted as workflows; we call this derived type of workflow ‘data logistics workflow’ or DL workflow. DL workflows can now be executed by a WfMS. This leads to the following effects:

- The executed DL workflows do not directly involve the clinical participants (like physicians). Thus, the strictness of workflow execution is eliminated. Instead of coordinating clinical participants, DL workflows have to exchange data between clinical applications. This happens transparently to clinical participants.
- The DL workflows take care that each clinical application is provided with necessary data on time; at least the data are moved between applications as fast as possible. Since this data communication task is enacted through (DL) workflows, the WfMS guarantees that it is performed correctly and

definitely. Here, the strictness of a WfMS can be regarded as a very positive effect.

Our approach is called *Process Based Data Logistics (PBDL)*. 'Normal' workflow management focuses on the ordered execution of work steps. Instead, PBDL focuses on data transport between applications. So the enacting WfMS enforces data exchange between applications which is desirable since the applications are provided with the right data at the right time. Clinical participants are not tackled by this, i.e. they are not directly controlled by a WfMS any more. This change of focus is explained in the following simplified example.

Patients with glaucoma must transmit some medical data like inner ocular pressure (IOP) and blood pressure to the clinic every few days (cf. Fig. 2). In order to do this the patients connect to a special (telephone system based) dialogue system (DS) by telephone (work step "record glaucoma data"). In a predefined dialogue the patients have to input their measured data. After the call – which is implemented as a step of a workflow – a subsequent work step must be executed that stores the transmitted data ("store diagnostic data") in the glaucoma register (GR). This step must be initiated and performed by an assistant medical technician since some additional administration related data must be completed. Would this process be implemented as a workflow, the assistant medical technician would be asked to execute the work step every time a patient was calling. This would not be acceptable for him since usually and practically he performs this kind of task once a day for *all* transmitted data of the day. This is much more efficient and effective from his perspective.

Due to the inapplicability of directly implementing the workflow, we extract data transport requirements out of the medical process. The data communication requirements can be characterized as follows:

- What has to be done?
  Move patients' data from the DS to GR
- When has it to be done?
  After a work step "record glaucoma data" was executed by a patient.

PBDL, unlike conventional workflow management, does not force the user to execute predefined steps, but supports the clinical users' work by transparently taking care of data communication. This change of focus can be nicely observed in Fig. 2. The upper scenario describes the workflow that reflects the scenario from above. Patients and assistant medical technicians would have to perform the two steps always in exactly the predefined order. The lower part of Fig. 2 shows how PBDL derives a DL workflow from such a process model. Here a work step "move data" that implements data transport between applications is used to communicate data between the two applications involved in the process, i.e. between the DS and the GR. It is specified what data have to be transmitted (here: IOP, blood pressure). It is also specified when this transmission must take place (here: incoming telephone call to DS); this latter triggers the execution of the work step, i.e. the transmitted patient data are automatically moved to the GR after the patient's call. Also we do not want to dwell on this in this paper, "move data" is a workflow template which is taken out of a PBDL library. This library contains typical work steps needed for PBDL.

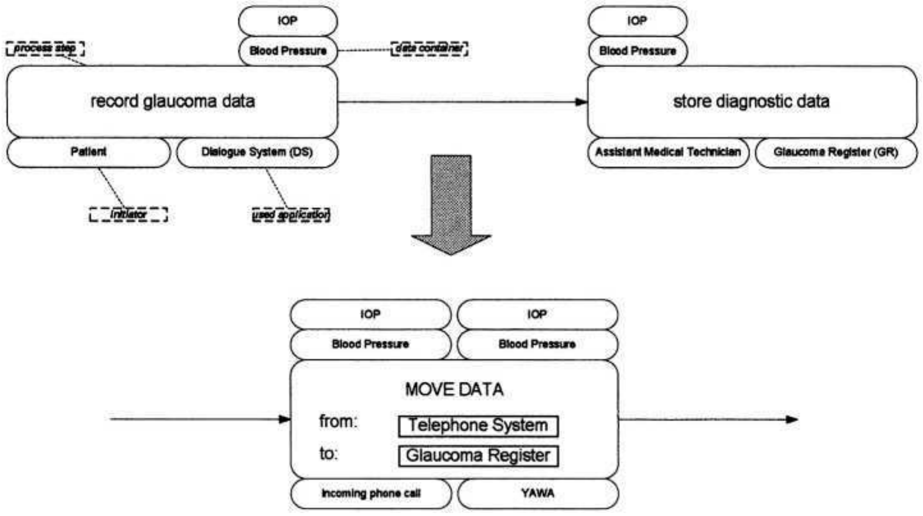


Fig. 2: Transforming a process model into a DL workflow

PBDL imposes system support in two areas:

- Medical processes have to be compiled into a data centric view, i.e. they have to be translated to DL workflows (cf. Fig. 2).
- Applications – involved into clinical processes – must be observed in order to recognize that a specific data transport step must be initiated (here: the DS application must be observed to trigger the "move data" DL workflow). When such an application execution is detected one or more steps of one or more DL workflows are triggered.

The DL approach is divided into four steps:

1. The clinical processes have to be recorded and documented.
   This step might be surprising; however, we need the clinical processes as base description of the application area.

2a. The clinical processes have to be transformed into a data centric format, i.e. into DL workflows.
   This transformation puts the data involved in a workflow into the center of interest.

2b. Involved applications must be wrapped.
   These applications must be made observable in order to detect what methods are performed by the participants.

3. The DL workflow must be executed.
   There is a variety of alternatives how to execute a data logistics process. In the clinical environment connection servers can be used in order to implement data exchange between applications [JLM+04]. They are triggered by rules (i.e. ECA rules) and deliver data from one application to the next one. The rules are fired, e.g. in wrappers which observe applications in order to know what working steps are currently under execution. However also WfMS can be used to execute DL workflows; this latter concept will be pursued in this paper.

The steps 2a and 2b can be executed in parallel since they are both preparing the execution of DL workflows. A DL workflow must implement two major tasks: data transformation and data transportation. The latter takes care that data are moved between applications and is defined by data access parameters and data quantification. The former is responsible for transforming data in such a way that the sending and the receiving applications can understand them. This directly leads to the three issues ontology, format and terminology. We shortly introduce these two concepts in the following, before we are going to explain them in detail in Section 3.2 and Section 3.3, respectively.

Data transport is described by a function that moves the content of a data source, the "out data container" (ODC) of a producing work step, to a data sink, the "in data container" (IDC) of a consuming work step – considering the issues (i.e. variables) AP (access parameters) and Q (quantification). Here, data transport is described with the example of Fig. 2.

$$f_{\text{transport}}\text{: } ODC_{\text{record glaucoma data}}{}^{*}(AP,Q) \rightarrow IDC_{\text{store diagnostic data}}{}^{*}(AP,Q)$$

Data transformation is described by a function that transforms the data of an ODC to the data of an IDC considering the issues (i.e. variables) F (format), O (ontology) and T (terminology). Again, we use the example of Fig. 2 to describe the function:

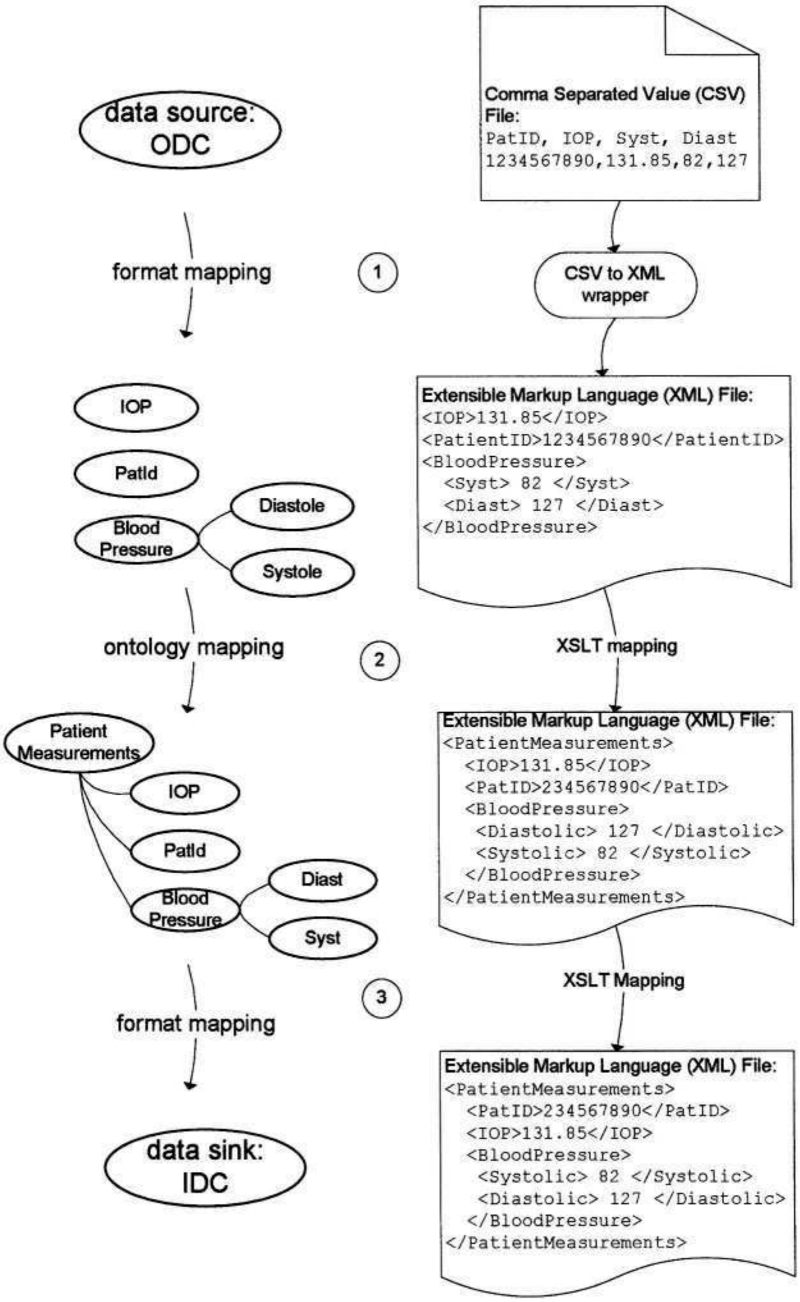
$$f_{\text{transformation}}\text{: } ODC_{\text{record glaucoma data}}{}^{*}(F,O,T) \rightarrow IDC_{\text{store diagnostic data}}{}^{*}(F,O,T)$$

Together, these two functions describe the necessary data logistics between two applications. A DL workflow step consists of a pair of these functions for every data flow that is derived from the clinical processes.

3.2 Data Transformation

In general the applications used in a workflow are based on different data models and so the data has to be transformed in order to be understandable by the applications mutually. For example the data container "blood pressure" in the dialogue system has no direct representation in the glaucoma register and so must specifically be transformed into one or more alternative notions. For instance, in the glaucoma register the blood pressure is described by a systolic and diastolic value. As mentioned in Section 3.1 three aspects have to be considered when transforming data: ontology, format and terminology. We propose a mapping process composed of three data transformation operations as illustrated in Fig. 3.

The ODC of the data source (here: the dialogue system) confirms to a given ontology, terminology and format (here: comma separated value (CSV) file). It has to be transformed into a suitable format for further transformations, which is accomplished by a *format mapping* step (cf. ① in Fig. 3). From there it can be mapped to the representation of the IDC of the data sink. In the example, a CSV-to-XML-wrapper translates the CSV file into an internal XML format. Only the format of the data is changed, not the ontology. The necessary conversion from the ontology of the data source to the ontology of the data sink is performed by an *ontology mapping* step (cf. ② in Fig. 3). In our scenario an XSLT [XSLT99] transformation is used. Finally the input format of the target system has to be generated by another *format mapping* step (cf. ③ in Fig. 3). Again we use an XSLT transformation to generate the XML input format for the glaucoma register as the data sink. In this example we omitted a terminology mapping step, which is in general necessary as well.



```
PatID, IOP, Syst, Diast
1234567890,131.85,82,127
```

```
<IOP>131.85</IOP>
<PatientID>1234567890</PatientID>
<BloodPressure>
  <Syst> 82 </Syst>
  <Diast> 127 </Diast>
</BloodPressure>
```

```
<PatientMeasurements>
  <IOP>131.85</IOP>
  <PatID>234567890</PatID>
  <BloodPressure>
   <Diastolic> 127 </Diastolic>
   <Systolic> 82 </Systolic>
  </BloodPressure>
</PatientMeasurements>
```

```
<PatientMeasurements>
  <PatID>234567890</PatID>
  <IOP>131.85</IOP>
  <BloodPressure>
   <Systolic> 82 </Systolic>
   <Diastolic> 127 </Diastolic>
  </BloodPressure>
</PatientMeasurements>
```

Fig. 3: The data transformation process

The mappings are performed in steps of a DL workflow. It may be useful to combine two or more steps in order to reduce the needed number of transformations; however, the underlying principles remain the same and reuse of transformation steps is not possible anymore. Ontologies are modeled using XML-Schema [XMLS01] and stored in a repository. They serve as input for a schema mapping procedure that generates the necessary XSLT style sheets, which are stored in a repository as well.

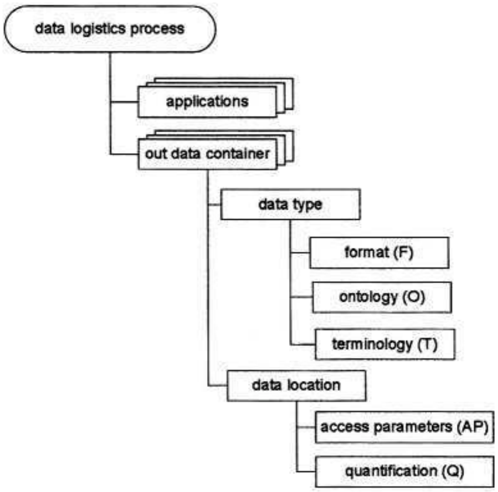


Fig. 4: Parts of a Data Logistics Process Step Definition

All the information needed to describe the three mappings (format, ontology, and terminology) must be derived from the original clinical process model. In order to accomplish this, data containers of work steps must include the following mandatory information (Fig. 4):

- The *data type* specification consists of the ontology (O), the format (F) and the terminology (T). The data type is needed for the data transformation steps.
- The *data location* specification consists of the access parameters (AP) and the quantification (Q). The data location is vital for data transport configuration described in the next section.

3.3 Data Transport

Before data can be transformed, they have to be retrieved from the source application. After data transformation the data have to be stored at the sink application. These tasks have to be supported by specifying the following information. The data location section of the data container description (cf. Fig. 4) holds the necessary information for this specification: the access parameters (AP) are system specific descriptions of how to access data, e.g. through a SQL connect string. The quantification parameter (Q) indicates which data to access, e.g. the attributes and values of an SQL query or the byte offset and length in a binary file. In order to reduce administration efforts we suggest storing such instance level configuration data in a repository as described in the next section.

While data transformation heavily needs application specific knowledge to be specified correctly, data transport is a more system oriented step. The former step needs domain specific knowledge to construct and interpret ontologies and terminologies while the latter is more relevant for computer system specialists in order to access data correctly.

3.4 DL Workflow Interpretation

The previous two sub-sections describe in general how data transport and transformation is constituted. In this section, we describe shortly how these tasks are facilitated in a real environment, i.e. in a real clinic.

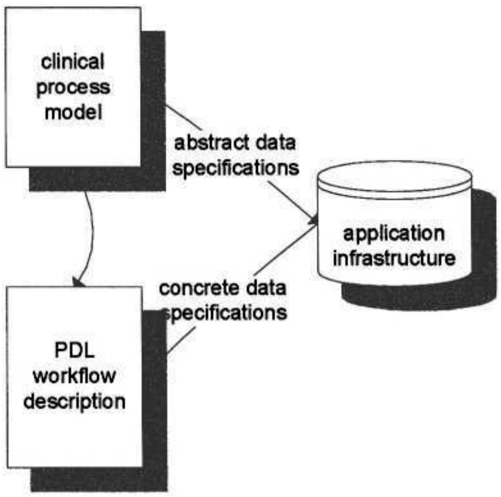

clinical process model
abstract data specifications
application infrastructure
concrete data specifications
PDL workflow description

Fig. 5: The creation of the data logistic description

To be executable in a real environment, the ODC and IDC of a DL workflow must be referred to concrete data of this environment. For instance, when a DL workflow has to be executed in the ophthalmology clinic in Erlangen-Nuremberg the data item “blood pressure” must be mapped to a specific attribute of a table in the clinic’s database. In another clinical environment, this data item must – for instance – be mapped to a specific field of a sequential file. Such a mapping must be made for each concrete execution environment. It is stored in a corresponding repository; in Fig. 5 this repository is called “application infrastructure”.

When a DL workflow is prepared to be executed in a concrete environment, the mapping information from the repository must be interpreted. This happens when the former clinical process is translated to a DL workflow (cf. Fig. 7). Since the mapping is done already, the transformation process can incorporate this information automatically.

Our concept is based on using abstract data descriptions in clinical process models. When a specific execution environment is selected, these abstract descriptions are replaced by concrete specifications. The separation between abstract and concrete specifications fosters the reuse of clinical process models and DL workflows, respectively. A kind of template process can therefore be reused elegantly in different execution environments.

4 A Case Study: Process Model to Data Logistics

In this section, we give an example, how to transform a clinical process model into a DL workflow description. To achieve this, we first describe the sample clinical process *self tonometry*. This process (cf. Fig. 6) is developed by and used in the ophthalmology clinic in Erlangen-Nuremberg.

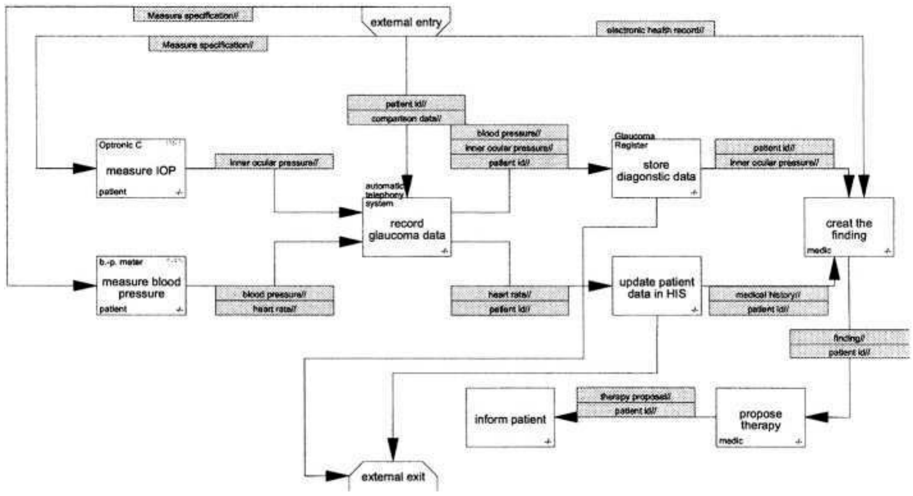


Fig. 6: The Self-Tonometry Process, as modeled in I>PM

The medical background is that for some glaucoma patients, the inner ocular pressure (IOP) of an eye has to be observed over a long period of time. This can be done by the patient himself. In order not to have to go to the clinic all the time, the patient has to transfer the measured eye data to the clinic. An automatic telephone system based on dial tone input is used to receive patient data. First, the patient has to be authenticated and than he enters his measurements. After the data has reached the telephone system, it has to be transferred to the patient's medical record in the hospital information system (HIS) and also into the glaucoma register (GR), where the science oriented data is stored. An ophthalmologist is observing the data, and orders a therapy or a more detailed examination when indicated by the measurements.

As described in the previous sections, we focus on the work steps "record glaucoma data" and "store glaucoma data". The data `blood pressure`, `IOP`, and `PatientID` have to be transferred from the telephone dialogue system to the glaucoma register. This should be done by a DL workflow.

4.1 Generating a Formalized Process Description

The medical process model depicted in Fig. 6 is not only described graphically but there is also a formal representation which is shown in the following. The process model is designed with a process modeler called *i>ProcessManager* [ProD03]. In the following we show some of the XML files generated in the context of the self tonometry process.

```
<PEK xsi:type="process" Name="record telephone call" ID="REP_ORTR(12720)">
    <Attribut Name="ParentName" Value="Self Tonometry" />
    <Attribut Name="ParentID" Value ="REP_ORTP(8348)" />
    <Attribut Name="Name" Value ="record telephone call" />
    <data_container_out Name="inner_ocular_pressure" ID="REP_ORTP(8350)" />
    <data_container_out Name="blood_pressure" ID="REP_ORTP(8318)" />
    <data_container_out Name="patient_id" ID="REP_ORTP(8340)" />
</PEK>

<PEK xsi:type="process" Name="store diagnostic data in glaucoma register"
ID="REP_ORTR(12675)">
    <Attribut Name="ParentName" Value ="Self Tonometry" />
    <Attribut Name="ParentID" Value ="REP_ORTP(8348)" />
    <Attribut Name="Name" Value ="enter IOP and blood pressure" />
    <data_container_in Name="inner_ocular_pressure" ID="REP_ORTP(8350)" />
    <data_container_in Name="blood_pressure" ID="REP_ORTP(8318)" />
    <data_container_in Name="patient_id" ID="REP_ORTP(8340)" />
</PEK>

<data_container Name="inner_ocular_pressure" ID="REP_ORTP(8350)" />
<data_container Name="blood_pressure" ID="REP_ORTP(8318)" />
<data_container Name="patient_id" ID="REP_ORTP(8340)" />

<FLOW ID="REP_ORTP(231678)">
    <From PEK="record telephone call" ID="REP_ORTR(12720)">
    <To   PEK="store diagnostic data in glaucoma register" ID="REP_ORTR(12675)">
    <data_container Name="inner_ocular_pressure" ID="REP_ORTP(8350)" />
    <data_container Name="blood_pressure" ID="REP_ORTP(8318)" />
    <data_container Name="patient_id" ID="REP_ORTP(8340)" />
</FLOW>
```

Fig. 7: Data logistics part of the process model

The XML representation of the medical process model has to be translated into a DL workflow, i.e. a data logistics specification. This is done via an XSLT script, which extracts process information and specifications of data flows and data containers. In Fig. 7, a fraction of the data logistic focused process model is depicted.

To execute the DL workflow steps, we are using a system called YAWA (Yet Another Wrapper Architecture) [LaLa03]. YAWA can be configured for data transport and transformation by a set of XML configuration files. The following (cf. Fig. 8) is a subset of the configuration files generated by our DL system.

In the following, the different sections for the YAWA configuration are explained

- `QUERYPROCESSOR`

  Specifies the context of the wrapper and the trigger. In this case, the `INTERVAL` specifies, that the data is queried each hour.

```
<?xml version="1.0" encoding="ASCII" ?>
<YAWASPEC xmlns:xlink="http://www.w3.org/1999/ xlink" >
<YAWA_ID>self_tonometry_transfer_IOP</YAWA_ID>
<QUERYPROCESSOR>
    <QUERYPROCESSOR_NAME>transfer_glaucoma_data</QUERYPROCESSOR_NAME>
    <QUERYPROCESSOR_CONFIG>
      <ENTITY>self_tonometry_transfer_IOP</ENTITY>
      <INTERVAL>3600</INTERVAL>
    </QUERYPROCESSOR_CONFIG>
</QUERYPROCESSOR>

<EXTRACTOR>
    <EXTRACTOR_NAME>get_glaucoma_data</EXTRACTOR_NAME>
    <EXTRACTOR_CONFIG>
       <TYPE>CSV</TYPE>
       <FILENAME>glaucoma_data.csv</FILENAME>
       <ATTRIBUTES>
           <PATID TYPE="Integer">
           <IOP Type="Integer">
           <SYST Type="Double">
           <DIAST Type="Double">
       </ATTRIBUTES>
    </EXTRACTOR_CONFIG>
</EXTRACTOR>

<RESULTPROCESSOR>
    <RESULTPROCESSOR_NAME>store_glaucoma_data</RESULTPROCESSOR_NAME>
    <RESULTPROCESSOR_CONFIG>
       <TYPE>XML</TYPE>
       <FILENAME>stored_glaucoma_data.xml</FILENAME>
       <TARGETS>
           <PATIENTMEASUREMENTS>
               <PATID TYPE="Integer">
               <IOP TYPE="DOUBLE">
               <BLOOD_PRESSURE TYPE="Record">
                   <SYSTOLE TYPE="Double"/>
                   <DIASTOLE TYPE="Double"/>
               </BLOOD_PRESSURE>
           </PATIENTMEASUREMENTS>
       </TARGETS>
    </RESULTPROCESSOR_CONFIG>
</RESULTPROCESSOR>

<TRANSFORMER>
    <TRANSFORMER_NAME>record_store_glaucome_data</TRANSFORMER_NAME>
    <TRANSFORMATION_CONFIG>
      <FROM SOURCE_NAME="get_glaucoma_data" ATTR_NAME="PATID" />
      <TO DEST_NAME="store_glaucoma_data" ATTR_NAME="PATIENTMEASUREMENTS/PATID" />
      <FROM SOURCE_NAME="get_glaucoma_data" ATTR_NAME="IOP" />
      <TO DEST_NAME="store_glaucoma_data" ATTR_NAME="PATIENTMEASUREMENTS/IOP" />
      <FROM SOURCE_NAME="get_glaucoma_data" ATTR_NAME="SYST" />
      <TO DEST_NAME="store_glaucoma_data"
         ATTR_NAME="PATIENTMEASUREMENTS/BLOOD_PRESSURE/SYSTOLE" />
      <FROM SOURCE_NAME="get_glaucoma_data" ATTR_NAME="DIAST" />
      <TO DEST_NAME="store_glaucoma_data"
         ATTR_NAME="PATIENTMEASUREMENTS/BLOOD_PRESSURE/DIASTOLE" />
    </TRANSFORMATION_CONFIG>
</TRANSFORMER>
```

Fig. 8: YAWA configuration for data transport

- `EXTRACTOR`
  Specifies the type of the data (CSV), the location (CSV filename) and the attributes to be extracted. This information is derived from the data container and the application infrastructure repository.
- `RESULTPROCESSOR`
  The same for the data sink. Both are responsible for the data transport.
- `TRANSFORMER`
  The transformer configuration describes the ontology mapping between the data source and the sink.

As we have mentioned, we can change the localization of data by changing the mapping between abstract and concrete data (cf. Section 3.4). This is a big advantage, since we can adjust process models to different execution environments smoothly.

5 Related Work

Closely related to process oriented data logistics is the concept of adaptive replication strategies in the health sector as presented in [NHHT02]. Basis of this system is the idea that communication in heterogeneous systems in certain cases may be boiled down to the replication of data. Nieman and Hasselbring describe a system that chooses the appropriate replication strategy based on predefined rule sets for the participating systems. It is pointed out that depending on the required degree of autonomy of a system synchronous or asynchronous replication strategies are preferable. In the case of asynchronous replication the need to synchronize arises and conflict resolution strategies have to be applied. The described adaptive replication manager takes care of this by supporting the user with a conflict manager component [NiHa02].

The adaptive replication manager is a sophisticated means of implementing communication between systems in the clinical environment. The system needs two kinds of information to run: Rules and the knowledge of the flow of information that should be managed. Both information are highly application domain specific and may be supported for example by process based data logistics systems.

6 Conclusion

In this paper an approach for Process Based Data Logistics in clinical environments is presented. Due to the complexity of clinical applications a model based approach is presented to cope with the complexity. Workflow management concepts are adopted to allow for an indirect process support, that provides clinical applications which necessary data on time. For this purpose process models are transformed into Data Logistics (DL) workflows considering both data transformation and data transport. For data transformation a format, ontology and terminology based approach is used. A case study illustrates the approach.

References

[BKMJ04] Beyer, M.; Kuhn, K. A.; Meiler, C.; Jablonski, S.; Lenz, R.: Towards a Flexible, Process Oriented IT Architecture for an Integrated Healthcare Network. Special Track on “Computer Applications in Health Care” (COMPAHEC 2004). Proceedings of the Annual ACM Symposium on Applied Computing (SAC’04), 2004; to appear

[DAB+01] Dolin, R. H., Alschuler, L., Beebe, C., Biron, P. V., Boyer, S. L., Essin, D., Kimber, E., Lincoln, T., und Mattison, J. E.: The HL7 clinical document architecture. J Am. Med Inform. Assoc. 8(6):552–569. 2001.

[EFGK03] Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A., 2003, The Many Faces of Publish/Subscribe, ACM Computing Surveys 35(2), 114-131, 06/2003

[HaMo93] Hammond, W. E., Moore, G. J. E. De (Eds.) et al.: "HL7: A Protocol for the Interchange of Healthcare Data.". Progress in Standardization in Health Care Informatics. IOS Press, 1993.

[HeSS96] Heinl, P., Schuster, H., Stein, K.: Behandlung von Ad-hoc-Workflows im MOBILE Workflow- Modell. In: Proceedings Softwaretechnik in Automation und Kommunikation - Rechnergestützte Teamarbeit, München, 1996.

[JaBu96] Jablonski, S., Bußler, C.: Workflow management - modeling concepts, architecture and implementation. London. International Thomson Computer Press, 1996

[JLM+04] Jablonski, S., Lay, R., Meiler, C., Müller, S., Hümmer, W.: Process Based Data Logistics in Clinical Environments. Proceedings of Multikonferenz Wirtschaftsinformatik (MKWI 2004), Track on Knowledge Supply and Information Logistics in Enterprises and Networked Organizations, 2004, to appear.

[LaLa03] Lang, M., Lay, R., 2003 : Yawa - Yet Another Wrapping Architecture. BTW 2003

[LaPH99] Lange, M., Prokosch, H., Hasselbring, W., 1999, Eine Taxonomie für Kommunikationsserver im Krankenhaus. Informatik, Biometrie und Epidemiologie in Medizin und Biologie 30 (1), 21-34, 1/1999

[NHHT02] Niemann, H., Hasselbring, W., Hülsmann, M., Theel, O.: Realisierung eines adaptiven Replikationsmanagers mittels J2EE-Technologie. In Proc. BTW 2003

[NiHa02] Niemann, H., Hasselbring, W., 2002, Adaptive Replikationsstrategie für heterogene, autonome Informationssysteme. In Tagungsband zum 14. GI-Workshop Grundlagen von Datenbanken, 2002

[ProD03] ProDatO GmbH, *i>ProcessManager,* http://www.prodato.de/software/processmanager, Retrieved 2003-12-03

[ReDa97] Reichert, M., Dadam, P., 1997. A Framework for Dynamic Changes in Workflow Management Systems. DEXA Workshop 1997

[XMLS01] XML Schema Specification, Version 1.0. http://www.w3c.org/XML/Schema, Retrieved 2003-12-03.

[XSLT99] XSL Transformations (XSLT), Version 1.0. http://www.w3c.org/TR/xslt, Retrieved: 2003-10-22.

Domain-Specific Concepts and Ontological Reduction within a Data Dictionary Framework

Barbara Heller*, Heinrich Herre#, Kristin Lippoldt*

Onto-Med Research Group
*Institute for Medical Informatics, Statistics and Epidemiology (IMISE)
#Institute for Computer Science, Department of Formal Concepts
University of Leipzig, Germany
Liebigstrasse 27, 04103 Leipzig, Germany
barbara.heller@imise.uni-leipzig.de

Abstract. This paper describes a new method for the ontologically based standardization of concepts in the medical domain. As an application of this method we developed a data dictionary which firstly focused on trial-specific context-dependent concepts and relations. The data dictionary has been provided to different medical research networks via the internet by means of the software tool *Onto-Builder*. It is based on an architecture which includes terminologies, domain-specific ontologies and the top-level categories of *GOL*[1]. According to our approach top-level concepts are used to build definitions of domain-specific concepts on a firm ground through the process of ontological reduction, which is currently under development and the main ideas of which are outlined in this paper.

1 Introduction

Traditional medical terminology systems, for example the International Classification of Diseases (ICD) and Systemized Nomenclature of Medicine (SNOMED), include concept bases in hierarchical structures. These concept bases are limited in their expressive power as a result of e.g. their static structure, undefined mixed partitioning criteria, a mixture of different views and imprecisely defined relationships. Taking into consideration these limitations it is clear that such terminology systems cannot be used as standards for developing medical software applications. The missing standards are one reason for singular solutions for software systems in medicine, e.g. management systems for patient data and medical administration, medical research data bases, hospital information systems, electronic health care records and clinical trial management systems. Each software system has its own data model derived mainly from the goals of the corresponding application. According to this background, new requirements on the representation methods of medical terminology

[1] General Ontological Language is a formal framework for building ontologies. GOL is being developed by the Onto-Med Research Group at the University of Leipzig [http://www.onto-med.de].

systems have arisen in the last decade. Engineers developing medical software applications are demanding reusability in more than one software application, the handling of multiple granularities, the management of multiple consistent views and so on to finally achieve semantic correctness in the data model. [1] [2] The realization of these criteria presupposes a solution to the real problems in medical terminology, such as the uncertainty of medical knowledge, context-dependent representation, clarification of different user views (e.g. corresponding to different branches), and the specification of relations between concepts. Therefore a deeper semantic foundation at the level of concepts is necessary. Our approach exploits the method of ontological reduction to achieve the semantic foundation of terminologies.

We have developed and implemented the software tool *Onto-Builder* which supports the internet-based construction of ontologically founded data dictionaries. Such a data dictionary is a terminological framework for domain concepts which is partly based on the top-level categories of GOL [3] [4]. The GOL (General Ontological Language) project was launched in 1998 as a collaborative research project of the Institute for Medical Informatics, Statistics and Epidemiology (IMISE) and the Institute for Computer Science (IfI) at the University of Leipzig. The project is aimed, on the one hand, at the construction of a formal framework for building and representing ontologies, and, on the other hand, at the development and implementation of domain-specific ontologies in several fields, especially in the medical domain [5]. The results of these research activities led in the establishment of the interdisciplinary research group Onto-Med (Ontologies in Medicine). The *Onto-Builder* is one of the self-developed applications of the Onto-Med research group which represents among others formal ontological aspects and its computer-based application in the biomedical domain.

Our paper is structured as follows. In section 2 we sketch how the data dictionary can be integrated into the development process of medical software applications. Following this, we introduce our methodology in section 3 and define the relevant components. Sections 4-7 provide deeper insight into our approach by describing the model of the data dictionary, introducing the relevant ontological categories and relations of GOL and discussing our idea of ontological reduction. In the last two sections we discuss the chosen method and outlook on further work in this area of ontological research.

2 Application Environment

Designing flexible and scalable software applications for the medical domain requires expertise in software engineering and knowledge modeling as well as experience in the medical domain and medical terminology. Especially the latter is often a problem in this area of software development because of the so-called acquisition gap, i.e. the unavailability of the necessary advanced domain knowledge. Furthermore, existing medical terminology systems like UMLS and SNOMED are not powerful enough to represent concepts in a machine-processable way. Therefore our aim is to provide a unified medical concept base which can be used by software engineers. We have developed a data dictionary which offers context-dependent definitions of concepts, and contains general concepts for medicine (e.g. `therapy`, `laboratory` `parameter`) in its basic configuration. For the design and implementation of new medical software applications, relevant concept definitions can be extracted and queried from the data dictionary. If no adequate definitions are available in the dictionary, the basic concepts can be expanded by means of appropriate alternative definitions.

The following figure gives an overview of the use of the data dictionary during the definition process of a data base.

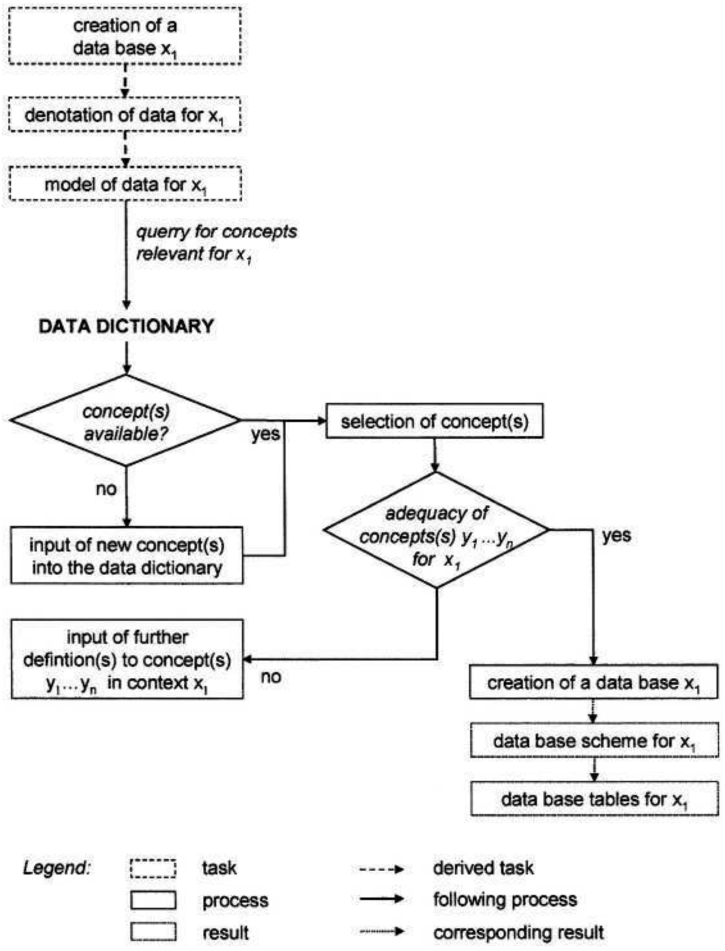

creation of a data base x_1
denotation of data for x_1
model of data for x_1
query for concepts relevant for x_1
DATA DICTIONARY
concept(s) available?
yes
no
selection of concept(s)
input of new concept(s) into the data dictionary
adequacy of concepts(s) $y_1 \dots y_n$ for x_1
yes
input of further defintion(s) to concept(s) $y_1 \dots y_n$ in context x_1
no
creation of a data base x_1
data base scheme for x_1
data base tables for x_1
Legend:
task
process
result
derived task
following process
corresponding result

Figure 1: Use of the data dictionary for creating data bases

3 Definitions and Methodology

Our approach to an ontologically founded terminology is based on different interacting computer-based components, namely terminology, data dictionary, domain ontology, and top-level ontology (see also fig. 2). In the following, we briefly define these components and describe their interaction within our ontological approach:

Terminology: According to [6] a terminology is the complete stock of the concepts, their definitions and names in a concrete domain. An example of a very early medical terminology in the area of anatomy is the *Nomina Anatomica* [7].

Data Dictionary: A data dictionary is to be understood as a collection of data which are described and interpreted as concepts with context. We claim that our notion of data dictionary is applicable on the one hand to different domains such as medicine, biology or technology, and on the other hand to different application scenarios such as paper-based documents or software applications.

Domain Ontology: We use the notion of a domain ontology in accordance with Gruber [8]. A domain ontology provides formal specifications and computationally tractable standardized definitions for the terms used to represent knowledge of specific domains in ways designed to enhance communicability with other domains.

Top-Level Ontology:. A top-level ontology is concerned with the most general categories of the world, their analysis, interrelations, and axiomatic foundation. On this level of abstraction ontology investigates kinds, modes, views, and structures which apply to every area of the world.

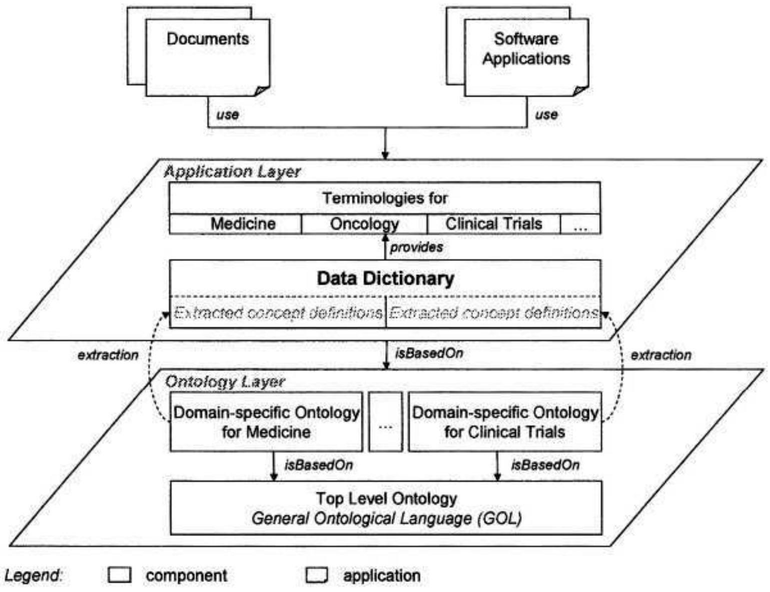


Figure 2: Two-layer model of an ontologically founded data dictionary

We assume as a basic principle of our approach that every domain-specific ontology (here in the field of clinical trials and medicine) must use as a framework some top-level ontology which describes the most general, domain-independent categories of the world. Therefore our data dictionary structure consists of two layers and is depicted in figure 2.

The *first layer* – called the application layer – contains two components: the data dictionary and the generic domain terminologies, i.e. domain-specific terminologies in the medical fields of oncology, clinical trials etc. The concept definitions of the generic domain terminologies are extracted from the identified and selected concept definitions of the data dictionary which are generic for the relevant domain. This domain generic information is taken as a basis for the definitions which are included in the component of generic domain terminologies. This means that these concept definitions are generic with respect to a confined area. The concepts of diagnosis, therapy and examination, for example, are defined generally in a terminology for medicine. In a special terminology e.g. for examination types, concrete specializations of general definitions are indicated with regard to single differentiable examination types, however.

The second component of the application layer consists of the data dictionary, which contains context-dependent concept definitions as well as references to corresponding information (e.g. radiographs, samples for medical documents) and provides the main definitions of concepts for domain-specific terminologies. The applications (here: documents and software applications) have access to the application layer from which they query relevant concept definitions and integrate them accordingly.

The *second layer* consists of two types of ontologies, namely the domain-specific ontologies (here for clinical trials, oncology and medicine) and the top-level ontology of GOL. The domain-specific ontologies describe formal specifications of concepts which are associated to a specific application. According to our approach, top-level concepts are used to build definitions of domain-specific concepts on a firm ground, and for this purpose we are developing a method of ontological reduction, the steps of which are outlined and discussed briefly in section 7. The GOL top-level ontology provides a framework with basic categories (e.g. universal/class, individual, quality, time, space, process and basic relations) which are described more precisely in section 5.

The two layers interact in the sense that the domain-specific concepts of the ontology layer are extracted from the data dictionary and are made available for the application-oriented concept descriptions which are provided for the application layer.

4 Application Layer

4.1 The Main Entities of the Data Dictionary

In this section we describe the model of the data dictionary and focus in particular on the following main entities: concept, denotation or term, description, context and relation. Definitions, relevant typings/classifications as well as references to the other components (Terminology, Domain Ontology, Top-Level Ontology) are included in the descriptions of these entities.

Concept, Denotation, and Term: A concept is an abstract unit of meaning which is constructed over a set of common qualities [6] and which can also describe a cognitive entity (e.g., feeling, idea, thought). A denotation or term consists of one or several words and is the linguistic representation of a concept [9].

In the data dictionary model we distinguish between generic (e.g., <disorder>, <process>, <treatment>) and domain-specific (e.g., <disease>, <symptom>, <medical treatment>) concepts. A generic concept has a general meaning in different domains due to its domain-independent qualities. The concept <treatment>, for example, generally expresses that something or someone is handled in a certain way. A concept is generic with respect to a class D of domains if it applies to every domain which is included in D. A domain-specific concept, however, has a meaning only in a certain domain. The concept <medical treatment> which is only relevant in the domain of medicine is an example of this kind of concept. A domain-specific concept of the data dictionary refers to at least one ontological category which is specific for this domain and which is included in the ontology related to this domain. The examples chosen also show that it is possible to change a generic concept into a domain-specific one by adding an attribute. Rules for changing a concept type, the composition and decomposition of concepts are the topics of a forthcoming paper [9].

Description: The description of a concept contains information about its meaning with respect to its qualities, its relations to other concepts, statements about its use, etc. [9].

Our model offers the possibility of handling alternative descriptions. There are various reasons for the occurrence of alternative descriptions, e.g. different levels of granularity, static/dynamic aspects, subject area-related specifications, organization-dependent or institution-dependent differences as well as different expert opinions due to medical facts which have not yet been investigated completely. These different alternative definitions are represented with the help of contexts.

Context: With regard to the various discussions on the notion of context, e.g., in [10], we give here the following preliminary definition: A context is a coherent frame of circumstances and situations on the basis of which concepts must be understood.

As in the case of concepts, we similarly distinguish between generic and domain-specific contexts. A context is – roughly speaking – generic if it is associated with concepts whose descriptions include general properties/qualities (e.g., a generic context is <process> which includes the concept <process course> with among others the generic property <process duration>). Contrary to this, a domain-specific context includes concepts whose qualities/properties and their corresponding values specifically apply to a given domain (e.g., a domain-specific context is <disease> which contains the concept <course of a disease> with among others the domain-specific property <course expression> and the values <chronic> or <acute>) [9].

Relation: According to [3], relations are defined as entities which glue together the things of the world. We distinguish between three classes of relations: basic, domain-specific and terminological relations [9]. Our method handles at the present stage 12 basic relations which are briefly outlined in section 5. Examples of domain-specific relations are: <treatedBy>, <SideEffectOf>, and for terminological relations: <synonymy>, <homonymy>, <polysemy>.

4.2 The Model of the Data Dictionary

A brief overview of the basic entities and relations of the data dictionary model is given in figure 3. The syntax of the model in figure 3 follows the UML[2] syntax, whereas rectangles represent classes (here: entities), rhombus n-ary associations (here: relations) and lines represent relations between the entities.

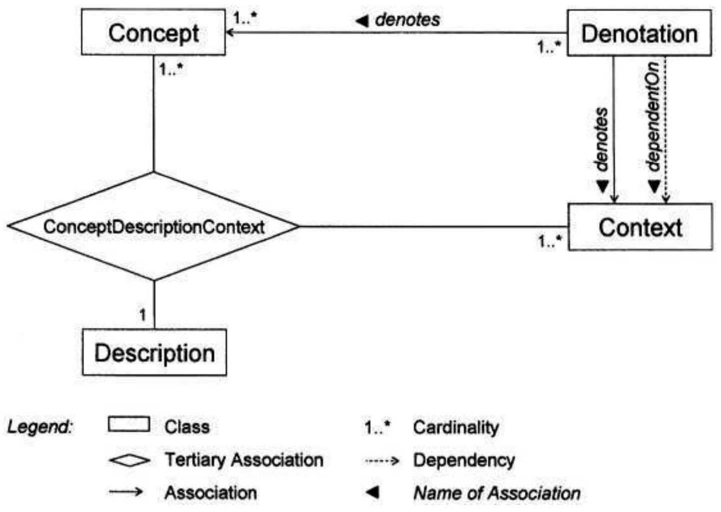


Figure 3: Data dictionary model (excerpt)

In our model, one `Concept` can be assigned to many `Description/Context pairs [1..n]` and one `Context` can be assigned to many `Concept/Description` pairs `[1..n]`. A `Concept` can be defined only by one `Description` in one `Context`. Different `descriptions` for a `concept` apply in different `contexts`.

The relation between `Description`, `Concept` and `Context` is expressed by the ternary association `ConceptDescriptionContext` which satisfies the abovementioned constraints. The entity `Denotation` describes `Concepts` and `Contexts` via the association *`denotes`*.

The dependency relation (here: *`dependentOn`*) between `Denotation` and `Context` means that the `Denotation` of a `Concept` can be dependent on the corresponding `Context`. If a `Concept` is not yet assigned to a `Context`, a default `Denotation` is given.

[2] Unified Modeling Language [11].

5 Ontology Layer

5.1 Domain-Specific Ontology

A domain-specific ontology describes a specification of basic categories as these are instantiated through the concrete concepts and relations arising within a specific domain. For this reason, ways must be found to take into consideration different experts' views on the domain concepts and relations, as well as different goals and contextually determined foci.

Domain-specific ontologies have a low portability; they can be transferred to other applications only to a very limited degree. Methods also have to be found to raise the degree of portability of domain-specific concepts, for example by using strictly modular description methods.

5.2 The Top-Level Ontology GOL

The General Ontological Language GOL is intended to be a formal framework for building and representing ontologies. The main purpose of GOL is to provide a system of formalized and axiomatized top-level ontologies which can be used as a basis for constructing more specific ontologies. The GOL-framework consists of three components representing different levels of abstraction. Meta-GOL contains basic principles of semantic choice, a general view on categories and classes, methods of semantic transformation, and principles of meta-ontological analysis such as consistency checking. GOL on the object-level consists of a basic logic and a representation language based on a typed logic which is specified by a syntax and a semantics. The core of GOL is a library of top-level ontologies [4].

In the following sections we sketch briefly certain ontologically basic categories and relations of GOL which support the development of domain-specific ontologies. A more detailed description of the ontological categories, the basic relations and some axioms of GOL are expounded in [3] [4].

Hierarchy of GOL Categories (Excerpt)
The following figure (Figure 4) shows an excerpt of the hierarchy of categories in GOL (advanced version) [3].

Sets, Classes, and Urelements
The main distinction we draw is between *urelements* and *classes.* C*lasses* (which include sets) constitute a metamathematical superstructure above the other entities of our ontology.

Urelements
Urelements are entities of type 0 which are not classes. Urelements form an ultimate layer of entities lacking set-theoretical structure in their composition. Neither the membership relation nor the subclass relation can reveal the internal structure of urelements. We shall assume the existence of three main categories of urelements, namely *individuals, universals,* and *entities of space and time.* An *individual* is a

single thing which is in space and time. A *(primitive) universal* is an entity that can be instantiated by a number of different individuals. We distinguish several classes of universals: immanent universals, concepts and textual types. We assume that the immanent universals exist in the individuals (*in re*) but not independently of them. On the other hand, humans as cognitive subjects conceive of (immanent) universals by means of concepts that are in their mind. For this reason, every relevant top-level ontology has to include the class of concepts. The symbolic-linguistic representation of concepts is based on textual types which exhibit another kind of universal. We want to emphasize that also higher-order (non-primitive) universals are needed for the classification of domain concepts (e.g. in the biomedical domain), and that a universal cannot be captured by its extension. For these reasons the underlying representation language of GOL contains (intensional) categorial types and (extensional) class types of arbitrary finite order. Here, in developing GOL, the Onto-Med team draws on its experiences in analyzing UML and other modeling languages [12].

Alongside urelements there is the class of *formal relations.* We assume that formal relations are classes of certain types.

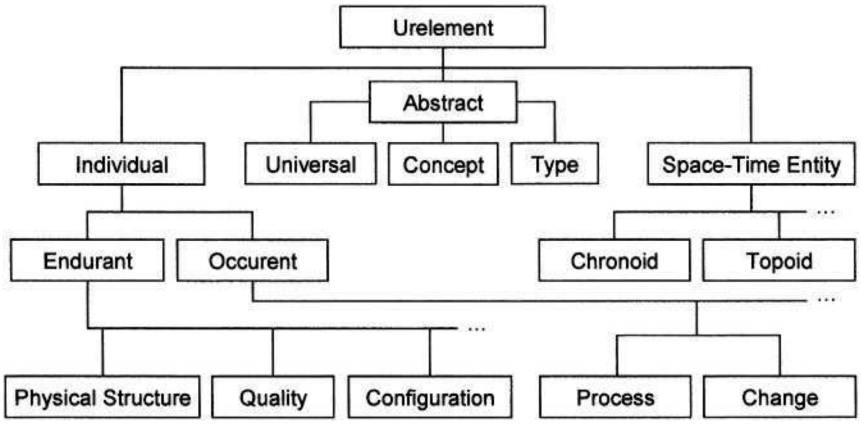


Figure 4: Hierarchy of categories in GOL (excerpt)

Space and Time

In the top-level ontology of GOL, *chronoids* and *topoids* represent kinds of urelements. *Chronoids* can be understood as temporal intervals, and *topoids* as spatial regions with a certain mereotopological structure.

Chronoids are not defined as sets of points, but as entities *sui generis.* Every chronoid has boundaries, which are called *time-boundaries* and which depend on chronoids, i.e. time-boundaries have no independent existence. We assume that temporal entities are related by certain formal relations, in particular the *part-of relation between chronoids,* the relation of *being a time-boundary ofa chronoid,* and the relation of *coincidence between two time-boundaries.*

Endurants and Processes

Individuals are entities which are in space and time, and can be classified with respect to their relation to space and time.

An *endurant* or a continuant is an individual which is in time, but of which it makes no sense to say that it has temporal parts or phases. Thus, endurants can be considered as being wholly present at every time-boundary at which they exist.

Processes, on the other hand, have temporal parts and thus cannot be present at a time-boundary. For processes, time *belongs to them* because they *happen in time* and the time of a process is built into it. A process *p* is not the aggregate of its boundaries; hence, the boundaries of a process are different from the entities which are sometimes called *stages* of a process.

Substances, Physical Structures, and Objects

Physical structures are individuals which satisfy the following conditions: they are endurants, they are bearers of properties, they cannot be *carried by* other individuals, and they have a spatial extension.

The expressions *x carries y* and *x is carried by y* are technical terms which we define by means of an ontologically basic relation, the *inherence relation* which connects properties to substances. Inherence is a relation between individuals, which implies that inhering properties are themselves individuals. We call such individual properties *qualities*. Examples of objects are `an individual patient, a microorganism, the heart` (each considered at a time-boundary).

We assume that the spatial location occupied by a substance is a *topoid* which is a 3-dimensional space region. A *physical object* is a physical structure with unity, and a *closed physical object* is a is a physical structure whose unity is defined by the strong connectedness of its parts. Objects may have (physical) boundaries; these are dependent entities which are divided into *surfaces, lines* and *points.*

Qualities and Properties

Qualities (or individual properties) are endurants; in contrast to physical structures, they are entities which can exist only within another entity (in the same way in which, for example, an individual form, color, role or weight exists only in a certain body). Examples of individual properties (qualities) are *this* color, *this* weight, *this* temperature, *this* blood pressure, this thought. According to our present ontology, all individual properties have in common that they are dependent on physical structures where the dependency relation is realized by inherence.

Situoids, Situations, and Configurations

Situations present the most complex comprehensible endurants of the world and they have the highest degree of independence among endurants. Our notion of situation is based on the situation theory of Barwise and Perry [13] and advances their theory by analyzing and describing the ontological structure of situations.

There is a category of processes whose boundaries are situations and which satisfy certain principles of coherence and continuity. We call these entities *situoids*; they are the most complex integrated wholes of the world, and they have the highest degree of independence. Situoids may be considered as the ontological foundation of contexts.

Relations

We can distinguish the following basic ontological relations of GOL in table 1, which are needed to glue together the entities introduced above. A more detailed description of the relations is given in [3] [4].

Table 1: Basic relations in GOL

Basic Relation	**Denotation(s)**	**Brief Description**
Membership	$x \in y$	set *y* contains *x* as an element
Part-of	*part(x, y)*	*x* is a part of *y*
	tpart(x, y)	*x* is a temporal part of *y*
	spart(x, y)	*x* is a spatial part of *y*
	cpart(x, y)	*x* is a constituent-part of *y (y contains x)*
	part-eq(x, y)	the reflexive version of *part*
	tpart-eq(x, y)	the reflexive version of *tpart*
	spart-eq(x, y)	the reflexive version of *spart*
	cpart-eq(x, y)	the reflexive version of *cpart*
Inherence	*i(x, y)*	moment *x* inheres in substance *y*
Relativized Part-of	*part(x, y, u)*	*u* is a universal and *x* is a part of *y* relative to *u*
Is-a	*is-a(x,y)*	x *is-a* $y =_{df} \forall u\, (u :: x \rightarrow (u :: y))$
Instantiation	$x :: u$	individual *x* instantiates universal *u*
	$x : y$	list x instantiates relation y
	$x ::_i y$	higher order instantiation, $i \geq 1$
Participation	*partic(x, y)*	*x* participates in process *y*, where *x* is a substance, an abstract substance or a substance process
Framing	*chr(x, y)*	situoid *x* is framed by chronoid *y*
	chr(x)	denotes the chronoid framing *x*
	top(x, y)	situoid *x* is framed by topoid *y*
	top(x)	denotes the topoid framing *x*
Location and Extension Space	*occ(x, y)*	substance *x* occupies topoid *y*
	exsp(x, y)	substance *x* has extension space *y*
Association	*ass(x, y)*	situoid *x* is associated with universal *y*
Ontical Connectedness	*ontic(x, y)*	*x* and *y* are ontically connected
Denotation	*den(x, y)*	symbol *x* denotes entity *y*

In table 1 the symbols x and y are entities. The concretization of the entities x and y depends on the type of the basic relation, e.g. tpart(x, y) means that x and y are *processes*. An exact specification of the admissible types of arguments of the basic relations in table 1 is presented in [4].

6 Ontological Reductions

An *ontological reduction* of an expression *E* is a definition of *E* by another expression *F* which is considered as *ontologically founded on a top-level ontology*. An expression is considered as ontologically founded on the top-level ontology GOL [14] if it is built up from atomic formulas whose meaning is inherited from the categories included in GOL. Ontological reductions exhibit a special case of semantic transformation. A semantic translation of a knowledge base *K* into a knowledge base *M* is a semantics-preserving function *tr* from the specification language *SL(K)* underlying *K* into the specification language *SL(M)* underlying *M*. Semantic translations can be used to compare the expressive power of ontologies and constitute an approach to the integration problem for knowledge bases. Semantic translations can be used as a formal framework for schema matching, which is a basic problem in many database application domains, compare [14]. An outline of this theory which is being elaborated by the Onto-Med group is presented in [15].

We sketch the main ideas concerning the notion of an ontological reduction based on a top-level ontology *GOL*. A definition *D* of a concept *C* for example is – usually – given as a natural language expression $E(C_1,..,C_n)$ which includes concepts $C_1,...,C_n$. The concepts $C_1,..,C_n$ are in turn defined by other expressions based on additional concepts. In order to avoid this infinite regress we select a certain number of concepts $D_1,.., D_k$ – which arise from *E* – as primitive. An embedding of $\{D_1,...,D_k\}$ into *GOL* is a function *tr* which associates to every concept D_i a category $tr(D_i) = F_i$ of *GOL* which subsumes D_i, i.e. every instance of D_i is an instance of $tr(D_i)$. The problem, then, is to find a logical expression E_1 based on $\{F_1,...,F_k\}$ which is equivalent to the initial expression *E*; such an expression is called an ontological reduction based on *GOL*. It may be expected that – in general – the system *GOL* is too weak to provide such equivalent expressions. For this reasons *GOL* has to be extended to the suitable system GOL_1 by adding further categories. GOL_1 should satisfy certain conditions of naturalness, minimality (the principle of Occam's razor), and modularity. The problem of ontological reduction includes four tasks:

1. construction of a set of primitive concepts (initialization problem)
2. construction of an ontological embedding into *GOL* (embedding problem)
3. construction of an extension GOL_1 of *GOL* (extension problem)
4. finding an equivalent expression (definability problem).

A developed theory of ontological reductions based on top-level ontologies is in preparation and will be expounded in [16].

7 Example

To illustrate some aspects of the ontological reduction method we consider the following short example. We focus on the first reduction step of selecting a set of primitive domain-specific concepts. Therefore, we will give a preliminary definition of primitive domain concepts.

Definition: A set of concepts C is called primitive concept base for a class DOM of domains (of the same granularity) iff every concept $d \in C$ is generic with respect to all domains in DOM and if there does not exist a concept $d \in C$ which is derivable from the set of concepts $C - \{d\}$ on the same granularity level.

A fully developed top-level ontology has to take into consideration levels of reality which include the problems related to the notion of granularity. The most important philosophical approach to this area of research - with respect to information system sciences is presented in [17] [18].

Application:
`Tissue` in the medical sense is to be seen as contained in a primitive domain-specific concept base because its meaning and interpretation is the same in different medical domains (e.g. pathology, endocrinology). The domain-specific concept `tissue` can be interpreted as a "part of an organism consisting of an aggregate of cells having a similar structure and function"[3]. Normally the concept `tissue` can be partly derived from the more granular concept `cell`. In our approach the derivation of concepts is limited to concepts of the same level of granularity and therefore the concept `tissue` is not derivable from the concept `cell`. In contrast to `tissue` the concept `fatty tissue` should not be considered as a primitive concept. It has the same meaning in different contexts but can be derived directly from the concept `tissue` and the concept `fatty` on the same granularity level. Further examples for primitive domain-specific concepts are `body, cell, organ, tumour, disease, therapy.`

To give an example for the main ideas of the ontological reduction sketched above we consider the

concept *C*	`organ system`
and its	
definition *D*	`A group of organs, vessels, glands, other tissues, and/or pathways which work together to perform a body function within a multicellular organism.`

As a first step we analyze the natural language definition *D* with regard to the concepts and relations it includes. These concepts and relations must be classified in primitive and derived concepts and relations. In the given definition the following concepts should be included, among others, in a primitive domain-specific concept base: `organ, vessel, gland, tissue, organism`. For further analysis let us consider the primitive concept `tissue` and focus on its structural aspects. The concept `tissue` has to be classified within the top-level ontology GOL as a substance. This assignment is part of the ontological embedding of the base of primitive concepts into the hierarchy of categories of *GOL*, i.e. `(tissue is-a substance)`.

Further steps of the ontological reduction have to take into consideration suitable extensions of *GOL* to finally achieve formal expressions (in the framework of GOL) which are semantically equivalent to the concepts included in the primitive concept base C.

[3] [http://www.hyperdictionary.com/]

8 Results and Discussion

With regard to the construction of a standardized terminology for software applications in medicine we have developed a methodology for an ontologically founded data dictionary. The methodology is based on two layers – the application layer and the ontology layer. The application components and theories at the two layers have been developed in parallel since 1999. One result of our work on the ontology layer is the development of the top-level ontology of GOL with approximately 50 basic categories and 12 basic relations. In the area of the domain-ontology we have started with the definition of domain-specific concepts which are partly based on top-level categories.

Concerning the application layer we have constructed a data dictionary for clinical trials which contains context-dependent concept descriptions. This data-dictionary has been implemented as the web-based software tool *Onto-Builder* [19]. This tool has been provided via the internet to several research networks with approximately 500 medical experts. Against this background, the handling of different expert views is indispensable within the *Onto-Builder*. This requirement is fulfilled with the availability of contexts in the data dictionary model which handle different expert views, granularity issues as well as special aspects of clinical trials. The present version of the data dictionary includes approximately 13 contexts, 1000 domain-specific concepts and 2500 concept descriptions.

Using the data dictionary, a higher level of harmonization of concepts and concept descriptions in different clinical trial protocols has been achieved. This has been possible due to the availability of a terminological concept base which has led in turn to an improved quality assurance in the clinical trial context.

9 Conclusion and Future Work

The evaluation of the application and theory components has shown that the underlying models of the data dictionary and the top-level ontology of GOL can be adapted to other domains and to other ontologies (e.g. DOLCE) [20].

At the present stage our data dictionary is merely a concept base for clinical trials and not yet fully based on domain ontologies. The reasons for this lie on the one hand in the extraction of domain-specific concept descriptions from the ontological layer which has not yet been realized completely. On the other hand it is connected to the problem of the ontological reduction of natural-language concept definitions via a semi-formal definition to formal propositions based on the built-in top-level ontology and its extensions. In our methodology we have already developed and partly integrated the first attempts at solving the ontological reduction problem.

Our future work consists in:

- the expansion of the theoretical framework with additional basic categories, e.g. situations, views and qualities
- the enlargement of the ontological reduction method according to functional aspects

- the elaboration of a theory of contexts and its evaluation in the area of clinical trials
- the incremental refinement of domain-specific concept descriptions with top-level categories
- the development of criteria for the specification of domain-specific concept types
- the explicit representation of semi-formal descriptions of domain-specific concepts
- the adaptation of the data dictionary to accommodate clinical trials in additional medical research networks.

10 Acknowledgement

We want to thank our medical and biometrics experts in the *Competence Network Malignant Lymphomas* (Grant No.: 01GI9995/0) and the *Coordination Centers for Clinical Trials, Cologne* and *Leipzig* for their fruitful discussions in the field of clinical trials and medicine. Many thanks are due to the members and Ph.D. students in the Onto-Med Research Group for implementing software modules for the *Onto-Builder* terminology management system and for the numerous discussions in building a top-level ontology. Last but not least we thank Evan Mellander for his efforts in the editorial work on this paper.

11 References

[1] Cimino JJ. Desiderata for Controlled Medical Vocabularies in the Twenty-First Century. *Meth Inform Med* 1998; 37(4-5):394-403.

[2] Rector AL. Clinical Terminology: Why Is it so Hard? *Meth Inform Med* 1999; 38(4-5):239-252.

[3] Heller B, and Herre H. Ontological Categories in GOL. *Axiomathes* 2004; 14:71-90.

[4] Heller B, and Herre H. Formal Ontology and Principles of GOL. Leipzig: Research Group Onto-Med, University of Leipzig; 2003. Report No. 1.

[5] Heller B, and Herre H. Research Proposal. Leipzig: Research Group Onto-Med, University of Leipzig; 2003. Report No. 2.

[6] Deutsches Institut für Normung e.V. DIN 2342 Teil 1: Begriffe der Terminologielehre. Berlin: Deutsches Institut für Normung e.V.; 10/1992.

[7] International Anatomical Nomenclature Committee. *Nomina Anatomica.* São Paulo; 1997.

[8] Gruber TR. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies* 1995; 43(5/6):907-928.

[9] Heller B, Herre H, Lippoldt K, and Loeffler M. Terminology Management for Clinical Trials (submitted).

[10] Bouquet P, Ghidini C, Giunchiglia F, and Blanzieri E. Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics* 2003; 35:455-484.

[11] Booch G, Jacobson I, and Rumbaugh J. *The Unified Modeling Language User Guide.* Amsterdam: Addison-Wesley; 1999.

[12] Guizzardi G, Herre H, and Wagner G. On the General Ontological Foundation of Conceptual Modelling. In: Spaccapeitra S, March S, Kambayashi Y, eds. 21st

International Conference on Conceptual Modelling (ER-2002); 2002 Oct 7 - 11; Tampere: Berlin: Springer; 2002. p. 65-78.

[13] Barwise J, and Perry J. *Situations and Attitudes.* Cambridge, MA, USA: Bradvord Books, MIT Press; 1983.

[14] Rahm E, and Bernstein PA. A survey of approaches to automatic schema matching. *The VLDB Journal* 2001; 10:334-350.

[15] Heller B, Herre H, and Loebe F. Semantic Transformation of Ontologies. forthcoming.

[16] Heller B, Herre H, and Loebe F. Ontological Reductions Based on Top-Level Ontologies. forthcoming.

[17] Poli R. The basic Problem of the Theory of Levels of Reality. *Axiomathes* 2002; 12:261-283.

[18] Poli R. Ontological methodology. *Int. J. Human-Computer Studies* 2002; 56:639-644.

[19] Heller B, Kuehn K, and Lippoldt K. Onto-Builder - A Tool for Building Data Dictionaries. Leipzig: Research Group Onto-Med, University of Leipzig; 2003. Report No. 3.

[20] Masolo C, Borgo S, Gangemi A, Guarino N, Oltramari A, and Schneider L. Wonderweb Deliverable D17. Preliminary Report, Version 2.0. Padova [Italy]: ISTC-CNR; 2002.

A Universal Character Model and Ontology of Defined Terms for Taxonomic Description

Trevor Paterson[1], Jessie B. Kennedy[1], Martin R. Pullan[2], Alan Cannon[1], Kate Armstrong[2], Mark F. Watson[2], Cédric Raguenaud[1], Sarah M. McDonald[2], and Gordon Russell[1]

[1]School of Computing, Napier University, Edinburgh, EH10 5DT, U.K.
{j.kennedy, t.paterson, g.russell, a.cannon}@napier.ac.uk
[2]Royal Botanic Garden, Edinburgh, EH3 5LR, U.K.
{m.pullan, k.armstrong, m.watson}@rbge.org.uk

Abstract. Taxonomists classify biological specimens into groups (taxa) on the basis of similarities between their observed features (‘characters’). The description of these ‘characters’ is therefore central to taxonomy, but there is currently no agreed model, defined terminology nor methodology for composing these descriptions. This lack of a common conceptual model, together with the individualistic working practices of taxonomists, means that descriptions are not composed consistently, and are not easy to interpret and re-use, nor are datasets comparable. The purpose of the Prometheus II project is to improve the interpretation and comparison of plant descriptions. To this end we propose a new conceptual model for unambiguously representing character descriptions, and have developed a controlled vocabulary as an ontology of defined terms, which will be used to describe specimens according to our character model.

1 Introduction: Problems with the Quality of Descriptive Data

Taxonomy is the branch of biology concerned with the classification of organisms into an ordered hierarchical system of groups (taxa) reflecting their natural relationships and similarities. The central taxonomic process, establishing relatedness and classifying organisms, is based upon the identification and description of variation between comparable structures on different specimens, with the critical taxonomic skill being the identification of such ‘characters’ that prove useful for classification. Integral to this process is the ability both to define the ‘character’ concepts used, and to describe the observed ‘character states’ precisely.

Whilst character data are the basic building blocks of descriptive data, there is little consensus amongst taxonomists on what the term ‘character’ actually means, making the interpretation of taxonomic descriptions problematic, nor is there an agreed terminology with which to compose descriptions. Specimen descriptions represent a huge potential data resource, not just for future taxonomic revisions, analyses and the creation of identification keys *etc.,* but for other biological disciplines such as biodiversity and ecological studies. However, these uses require the meaningful integration of data from different description sets, which, in the absence of both an agreed character model and particularly a shared descriptive terminology, is currently not possible.

A character concept is derived during the taxonomic process by partitioning observed variation into characters and 'character states' i.e. the combination of a structure, the aspect of the structure being described (its property) and its possible states [1,2]. For example, a 'character' *leaf shape* may be recognized and, in a group of specimens, the 'states' *obovate, ovate* and *oval* observed. The description of that group of specimens would then read 'leaves obovate, ovate or oval'.

A 'character' might be defined in general terms as 'a statement on a feature of the organism' although many different specific definitions have been proposed [3]. Diederich *et al.* [4] propose a useful universal definition of 'character', decomposed into structure, property and score, which enables taxonomists to be explicit about every aspect of a character statement. However, Diederich did recognize that in many usages of'character', the 'property' is not explicitly recorded.

Taxonomic descriptions are composed of descriptions of character states for an individual specimen or a group (i.e. a taxon, such as a particular species, genus *etc.*). Traditionally descriptions are recorded in semi-formal natural language, and several electronic description formats and applications have been developed to allow the storage and analysis of data [5-7]. However, these formats have been developed to support the flexible use of character concepts and terminology. Flexibility implies a lack of standardisation in the use of character concepts, and in the absence of a well-defined character model and an agreed terminology, descriptions are generally only consistent within a single data set. Consequently, taxonomists cannot communicate the basis of their work adequately [8], nor meaningfully integrate data from various sources.

To date it has not been possible to achieve universal definitions for characters or the terminology used to describe 'characters' (see for example the experiences of TDWG who have attempted to standardize the terminology for botanical descriptions [9]). This is not surprising given the wide variation in structures and characters across the whole taxonomic range (e.g. comparing algae with flowering plants). Furthermore, descriptive terminology is domain specific, with the same word having differing meanings in different taxonomic groups (homonyms), or different words being used in various taxonomic fields to describe the same concept (synonyms).

The Prometheus project [10] aims to improve the methodology for taxonomic description and provide tools for recording data more rigorously. Taxonomists recognize problems with current working practice, and several authors have suggested that there should be a standard approach to taxonomic description [4,9,11], however, they are concerned that an improved methodology should not restrict the expressiveness of their descriptions. Prometheus aims to provide an integrated suite of tools for developing descriptive ontologies, automating the generation of proformas (description templates, detailing the 'characters' to be scored for a specimen), and providing interfaces for entering and storing descriptions to a database that will form a repository of compatible descriptive data. A prototype ontology has been developed which defines and constrains terms necessary for describing flowering plants (angiosperms).

2 Representation of Characters as 'Description Elements'

We propose a new data model for character description, which facilitates the standardization and integration of data. The model is intended mainly for recording the information collected for new descriptions, but might also be used to record an interpretation of an existing description. The creation of a defined terminology with which to compose actual descriptions is addressed in section §3.

To allow taxonomists to be explicit about every aspect of a character statement we have developed Diederich's definition of 'character' [4], which he decomposed into structure, property and score (Fig 1.). This composition of character is represented as a Description Element (DE), in which a character description is created by recording the defined structure, the defined property and the observed score (Fig 1.). However, we note that taxonomists record both quantitative and qualitative data and that whilst this decomposition is readily applicable for quantitatively measured characters with a real score (such as the properties *length, width, height etc.*), many qualitative statements record the 'state' of a structure as the score, and often the associated 'property' is less readily discernable, and typically not explicitly recorded. To accommodate this, the model requires two kinds of DEs: Quantitative and Qualitative, where Qualitative DEs do not require explicit association of a property with the structure/state combination (Fig 1.). *Specimen Descriptions* are composed of the set of DEs for that specimen.

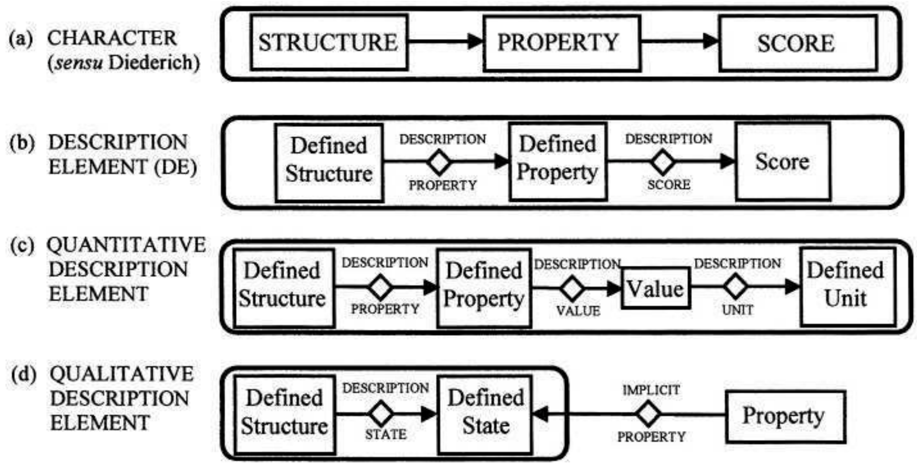


Fig. 1. Diederich's decomposition of 'character' into structure, property and score is represented as a Description Element, which uses defined terms to capture each element of a character statement. Quantitative DEs include a numeric value as score and require a defined unit (e.g. Leaf Length 5 mm); Qualitative DEs do not explicitly record a property, but states describe an implicit property (e.g. Leaf Oval *(OutlineShape)*). The recording of multiple values or states within a DE is discussed later (§2.3).

2.1 Quantitative Description Elements

In order to record the ‘character’ statement ‘leaf length 5 cm’ a quantitative DE is composed specifying a defined structure (*leaf*), an explicit defined property (*length*), a value (an individual number: 5) and the appropriate defined unit *(cm).* For quantitative statements that do not have units, e.g. *number of petals*, ‘*count*’ is defined as a unit. Clearly there is a finite list of defined quantitative properties which can be described by these elements, which might minimally consist of {*Angle, Density, Diameter, Height, Length, Number, Width*} and be expanded to allow further defined quantitative properties as needed (e.g. *Colour*, as defined by RGB values *etc.*). Detailed ontologies defining measurement concepts (i.e. units and dimensions *etc.*) for the biological domain are being developed by others (e.g. [12]) and could ultimately be used to constrain and define the terms used in Prometheus Quantitative DEs.

2.2 Qualitative Description Elements

In order to correctly record a statement such as ‘*leaves oval*’ a qualitative DE is composed with a defined structure (*leaf*) and a defined qualitative state (*oval*). Note that no explicit property is specified for qualitative scores, although a state is associated with an implicit property, which might be defined by grouping states into ‘usage groups’ (see section §3.2).

Arguably it should be possible to describe all physical data quantitatively, and Prometheus would encourage quantitative description where practicable to permit the direct comparability of DE data. However, often this is neither reasonable nor useful to taxonomists, who assign qualitative states by categorising continuous quantitative variation or represent complex character properties with more easily handled discrete states. The detail required to describe such states in absolute quantitative terms would often be prohibitive. For example, leaf shape is usually described in terms of discrete states such as *linear* or *lanceolate,* although in reality leaf shape is a continuum.

2.3 Representing Concrete and Abstract Data

When describing a specimen, character data may be an accurate record of the state of an actual individual structure, or may represent an average or representative state for the collection of such structures on the specimen. Taxonomists use both types of data but often do not distinguish between them. In order to distinguish between the former and latter case, a DE can be explicitly recorded as Concrete or Abstract. Some taxonomic work will represent variety by recording a large number of instances of concrete DEs for a structure/state, whereas other work will express variety as a collection or range of abstract DEs. The types of analyses that can be performed on description data will depend on whether the data is real (concrete) or summary (abstract). Taxon descriptions are by definition abstractions as they are a summary of the specimens in the taxon and will only be composed of abstract DEs.

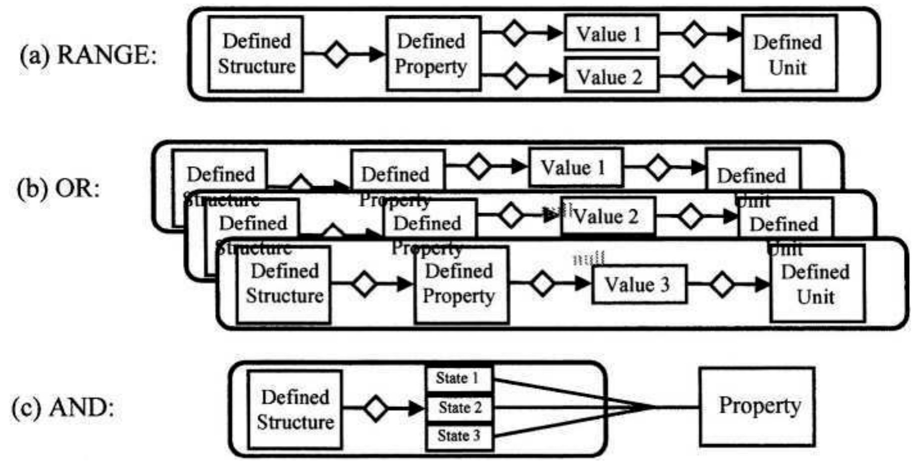


Fig. 2. Capturing Variability in Description Elements. (a) Quantitative ranges are captured by storing two values in a quantitative DE (e.g. Leaf Length 5 to 10 mm). (b) Multiple DEs describing the same structure and property capture multiple single-valued alternative states (OR) (e.g. Petal Number 4 or 8 or 12). (c) Multiple states of a single property can be saved in a single qualitative DE (AND) (e.g. Leaf Brown and Green and Yellow *(Colour)*).

2.4 Recording Variability in Description Elements

It is common practice for a description to record a range of values for a given measurement (e.g. *length 5-10mm*). The frequency with which recording ranges is necessary makes it sensible for our implemented model to allow a quantitative DE to explicitly record a pair of score values to capture a range (Fig.2a). However, an observed range is not necessarily a continuum, for example if the flowers on a specimen may be observed to have 3, 5 or 7 petals. In this case the ‘range’ can be represented by recording multiple alternative (OR’ed) DEs for that property (Fig. 2b). Only abstract DEs will ever express ranges or alternatives, as concrete DEs record actual measurements for a single, real structure.

Taxonomists also currently record ranges in qualitative states, for example *leaves round to ovate.* It is not possible to unambiguously interpret such a description, as there is no representation of intermediates in the categorized continuum of the character states. Therefore any interpretation of a range is subjective. If it is not possible to record a range quantitatively it is necessary to represent the range of possible qualitative states by defining states that break up the continuum of variation without leaving significant gaps, and to record the existence of multiple alternative (OR’ed) DEs for that property (similar to Fig.2b). For example, a specimen may have leaves with apices that range between *acute* and *acuminate.* Ideally this would be recorded quantitatively as a range of angles (e.g. 10-80°). However, this range could also either be defined by two qualitative states which encompass a wide range themselves (leaves with an apex angle of <50° are *acute*; with an apex angle of >50° *acuminate*), or a user could define a large set of states that describe all the possible intermediate angles. The leaf apex is then described with a set of DEs that explicitly describe all the possible variations.

Whilst (abstract) quantitative DEs can record ranges, or alternative values, for a quantitative property, a single quantitative DE can only have one measurement for one property (e.g Leaf: Length: 5 mm; without considering measurement accuracy here). On the other hand, because qualitative states are not simple quantitative measures and are not necessarily mutually exclusive, qualitative descriptions may include multiple states ('AND' Fig. 2c) (e.g. Leaf: Brown and Green and Yellow). A qualitative DE can therefore record multiple (AND) states for a given score, whereas alternative (OR) states are recorded in multiple linked DEs for separate instances of a structure, with each DE describing the same implicit property (e.g Leaf: Brown OR Leaf: Yellow OR Leaf: Green).

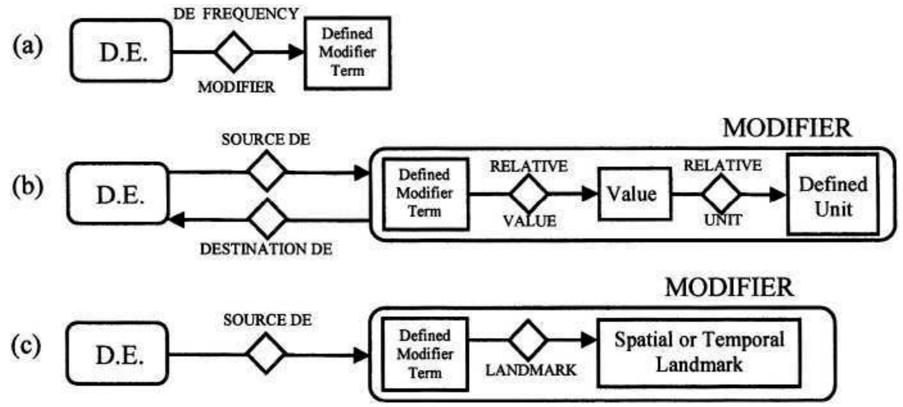


Fig. 3. (a) Description Elements (DEs) can be modified by a simple frequency modifier term. (b) Two DEs can be related by Relative, Spatial or Temporal Modifiers (relative modifiers may also include a value and unit). (c) DEs may also be modified by landmark statements.

2.5 Modifiers of Description Elements

To allow rich and flexible description using our model it is necessary to allow additional information to be associated with DEs, this is achieved by using 'modifiers'. The simplest modifiers are *frequency modifiers*, which are simple defined terms that can be added to an abstract DE to indicate relative occurrence {e.g. *mostly, often, usually, sometimes, rarely*} (Fig.3a).

Other modifiers are required to relate two DEs in order to capture one state in relation to another, (e.g. *leaf length* in comparison to *leaf width*). These modifiers therefore link source and destination DEs, and have associated defined terms and possibly values (Fig. 3b). We distinguish three types of these modifiers, with associated sets of defined modifier terms: *Relative:* {*greater-than, less-than, equal-to, ratio, not-equal-*

to, less-than-or-equal-to, greater-than-or-equal-to}; *Spatial:* {*at, above, below, between*}; *Temporal:* {*after, before, while*}. Relative modifiers allow undefined scores to be related (e.g. *leaf length* 'less than' *leaf width*) or with an associated value (e.g. length is twice width: *length* 'ratio: 2' *width*). Spatial Modifiers allow measurements to be more accurately defined (e.g. *trunk diameter* 'at' *branch*. In order to allow the flexibility of natural language descriptions these modifiers can also relate a DE to a 'landmark statement', for example *trunk diameter* 'at' <breast height> (Fig.3c). Temporal modifiers allow the time of year, or sequential order of events to be recorded, and can again use 'temporal statements', e.g. *flowers* 'in' <spring>; *fruit colour* 'before' *fruit colour* (i.e. to describe unripe before ripe).

Whilst these modifiers allow storage of rich data in a less ambiguous form, reflecting the current style of natural language/free text description, actually processing and analysing some of this data in comparisons might prove highly complex, especially if landmark statements are included, which are essentially free text.

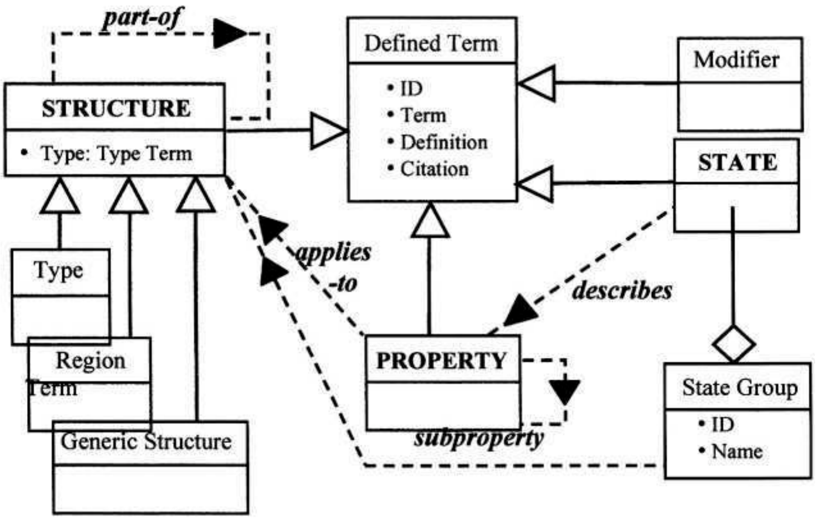


Fig. 4. Concepts and relationships in the descriptive term ontology. All terms are specializations of Defined Term. Structures can be 'Part-Of' other structures recursively, and may have attribute: Type (itself a specialized Structure). States are composed into groups, which may be restricted to ('applies-to') certain structures. Therefore these state groups may represent 'de facto' properties, which may include a structural context. Alternatively states can be considered to describe a given property, which may be applicable to only certain structures.

3 Using an Ontology to Specify a Defined Terminology

Taxonomists expressed concerns that using an ontology to define and constrain term usage in specimen descriptions might restrict the flexibility and expressiveness of current natural language description (which is, however, not machine processable). However, a minimal requirement for enhancing the interoperability of specimen descriptions is the consistent use of a set of defined terms. Capturing definitions and information about how terms can be used in relation to each other represents the creation of a semi-formal ontology, i.e. a constrained and structured form of natural language. Ontologies can be used to mediate both structural and semantic data integration by generating a unified view of local ontologies (as a mediated schema), and developing a common global ontology integrating concepts amongst data providers [13,14]. By specifying a standard controlled vocabulary for specimen description Prometheus will prevent semantic heterogeneity between descriptions that have been composed solely with defined terms from this common ontology. The relationships and classes within the Prometheus ontology are shown diagrammatically in Fig. 4. Instances of the primary classes of concepts: defined Structure, State and Property are used in Description Elements to create character descriptions according to our character model.

3.1 Structure Terms in the Ontology

The creation of a consensual ontology defining structural terms for the limited domain of flowering plants was chosen as a realistic initial goal for the project, particularly when restricted to the inclusion of macroscopic anatomical-morphological features found in traditional specimen descriptions. This ontology might subsequently be expanded to include further structural terms (e.g. microscopic and subcellular terms).

The ontology requires definition of the structure terms necessary to describe angiosperms. However, it is important that a description captures not only the structure being described, but its structural context and composition i.e. what it is *part of,* and what structures are *part of it.* These potential relationships between structures in the ontology are captured with a 'Part-Of' relationship (Fig. 4).

Rather than creating a universal structural 'map' of an idealized angiosperm, the taxonomists required that Part-Of relations in the ontology reflect the variety of possible structural compositions found across the taxon. This requirement is met by allowing a given structure to be defined as *potentially* Part-Of several other structures. Only when one of these contexts is chosen and used in an actual proforma or description will that particular structural context be affirmed. The Part-Of hierarchy therefore forms a Directed Acyclic Graph, but can be viewed more intuitively by the taxonomists as a branched tree with multiple instances of some structures (Fig. 5a,b). For example, *androecium* appears as part of five structures in the hierarchy and each of these structures can have two structural contexts, dependent upon whether florets are present, giving ten possible context paths that can be chosen for *androecium* (Fig.5c). Each path uniquely identifies the 'node' in the structure hierarchy, and allows a description to unambiguously specify the described structure both as a defined structure term and by its relationships to other defined structures. Defining terms in such an ontology, which specifies structural relationships between parts, therefore improves our character model, allowing specification of a defined structure in its context.

There are certain anatomical structures (e.g. *hairs, pores;* referred to here as 'Generic Structures') that might potentially be part of many if not most other structures. Similarly any structure can be subdivided into 'Regions' (e.g. *base, apex*). If Part-Of relationships were explicitly recorded for these structures there would be an unmanageable explosion of possible context paths in the ontology. For clarity we decided that Regions and Generic Structures represent specialized types of structures that are not explicitly included in the hierarchy, e.g. the region *base* has not been added to every structure in the ontology. Instead it is the responsibility of a user of the ontology to explicitly specify where these structures should be added to an instance of the ontology for use in descriptions (e.g. adding *hairs* to *leaves* and *petals,* and *apex* to a *leaf* when defining a proforma ontology, see §3.3).

A further specialized subclass of structure terms is 'Types'. Structure terms can be defined as a 'Type-Of' another structure term if they are examples of the supertype that always have a number (more than one) of descriptive states true for each instance of that supertype structure. Types reflect an awkward apparent blurring between states and structures when describing specimens, e.g. a *berry* is clearly a structure in itself, but it is also a collection of states for a particular structure (a *fruit* that is always fleshy, indehiscent and has seeds submerged in pulp). We exclude types from the structural hierarchy and treat them as an attribute of their supertype, so that in a description a fruit can be recorded as being of type berry.

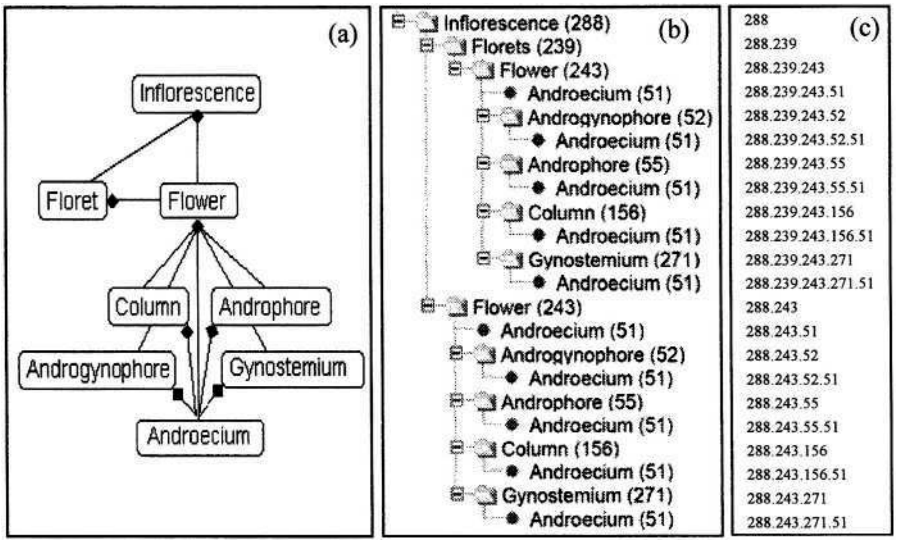


Fig. 5. Representing the 'Part-Of' hierarchy as (a) a Directed Acyclic Graph (b) a Tree Graph (c) a dot separated string of structure IDs detailing the hierarchical path.

3.2 State Terms in the Ontology

Taxonomists perceived the selection of allowed state terms for the ontology to be more problematic than specifying allowed structures. One objection was that individual taxonomists use their own personal preferred state terms and have an individual perception of state definitions and how they relate to other states. It is the aim of Prometheus to adequately define state terms to ameliorate these individualistic working practices. Another objection to prepopulating the ontology with state terms was that this is counter to taxonomic practice where taxonomists create their concepts of extant characters only by examining the specimens. Creation of a defined term list might imply predefinition and restriction of allowed character states, a criticism of other description formats. However, in the Prometheus model defined state terms are not character definitions, but are part of the vocabulary used to compose character descriptions at the time of specimen description.

As discussed previously (§2.2) it is difficult to define in a consistent and non-arbitrary fashion the underlying 'properties' associated with qualitative DEs and state terms. This is because a taxonomist's interpretation of a state can include aspects of several properties. Initial attempts to categorize state terms in terms of the qualitative property described (e.g. {*Arrangement, Colour, Shape, Texture etc.*}) suggested that for many state terms such divisions were arbitrary and contentious. For example, whilst 'red' is clearly a state *of colour*, is 'keeled' a *shape* or an *arrangement* of petals?

However, taxonomists can intuitively organize state terms into sets that are used to describe alternative aspects of the same feature (i.e. usage groups). The composition of such a set of states (a 'State Group') could be considered to circumscribe an implicit, *de facto* 'property', which in some cases is qualified by a given structural context. This reflects taxonomists' apparent conceptualization of the qualitative property of a 'character' as a gestalt of property in context of the structure being described.

Once created, analysis of these state groups reveals a meaningful 'property' that each group describes, and any particular structural context represented by the extent of the group. Thus a hierarchy of properties and subproperties can be created, which allows all state groupings to be defined in terms of described property and a possible structural context. For example we can distinguish different subclasses of *Arrangement:* e.g. *Architecture*, *Form*, *Position etc.* Furthermore a state group may be defined by a structural context of a particular property, for example all the states describing 'leaf' architecture. If expressed explicitly these properties would allow qualitative descriptions to be represented in a similar fashion to quantitative descriptions, with all states being the score of an underlying qualitative property, analogous to a value being the score of a quantitative property.

In our ontology groups of states are restricted in their usage to describe certain allowed structures; this is captured by the 'applies to' relationship (Fig.4) between properties (or state groups) and structures. However, some groups of states or properties (e.g. '*Textures*') can apply to such an extensive range of structures that it is not sensible to restrict usage to a subset of structures.

Ideally state groups would be composed of the set of exclusive alternative states for the property of a given structure (i.e. only one state can apply to an individual structure). However, the extent of such exclusive groups proved difficult to define such that a given instance of a structure would never be described by more than one state in a group or property. All state groups are therefore considered as potentially 'multistate' for a given description instance. Where state terms appeared to belong in more than one state group, it was apparent that the contextual meanings of the terms are not identical and in such cases it is necessary to create homonymous terms (with different definitions), which belong to separate state groups.

As the state term lists were being compiled, it became apparent that a large number of commonly used terms merely expressed the presence/absence of a structure (e.g. *stipulate*: possessing stipules), or enumerated a structure (e.g. *biovulate*: containing 2 ovules). In order to improve compatibility Prometheus aims to create more explicit, quantitative descriptions. As such there are explicit mechanisms to record presence or absence, and to count structures: therefore use of these types of state terms is discouraged. This becomes problematic where the state descriptors both imply presence of a structure, and the state of that structure, often where the implied structure is a region or generic structure (e.g. *tomentose*: densely covered in short hairs). The interpretation of such a state is contextual, and although *densely* and *short* could be quantitatively defined, it is impossible to define them acceptably for all contexts, thereby making it impossible to define *tomentose* quantitatively. Therefore although we could consider terms such as *tomentose* to have structural and quantitative dependencies we allow their use and assume that they are comparable across descriptions (in terms of the definition), leaving any discrepancy in the exact definition of *tomentose* to be resolved by the taxonomist where necessary.

3.3 Defining Proforma Ontologies

Our ontology aims to include all of the defined structure and state terms necessary to create DEs describing a given angiosperm specimen. Usage of the defined terms is constrained by a number of relationships specified in the ontology. States are grouped into usage groups, which may be linked to the structures which they are allowed to describe; structure types are identified for some structure terms; and a structural composition hierarchy has been specified which details all possible structural contexts. However, generic structures and regions are not yet specified in this hierarchy. In order to create an instance of the ontology that is actually used for a description set (i.e. for a description proforma) the user will select the structures (in context) he wishes to describe and which properties he wishes to describe for these structures. The taxonomist may additionally restrict the list of qualitative states that are available for a given structure. Part of this process involves explicitly adding the regions and generic structures that are to be described to the existing structural hierarchy. These processes, which both extend and restrict the parent ontology, can be seen to create a proforma specific ontology, for a given description set (Fig.6).

Descriptions composed using a specific proforma ontology are automatically compatible with other description data composed using any other proforma ontology derived from the same parent ontology, as the term definitions and structure path contexts are consistent.

A further complication with the specification of proforma ontologies, not considered in detail here, is that it may be necessary to represent any given structure node with more than one copy or clone. This will be necessary to represent multiple versions of a structure, for example when a taxonomist requires to distinguish multiple versions of a leaf that will be described separately (e.g. if there were two identifiable leaf forms present on specimens, some which tended to be small, brown and hairy, and others that were large, pink and glabrous).

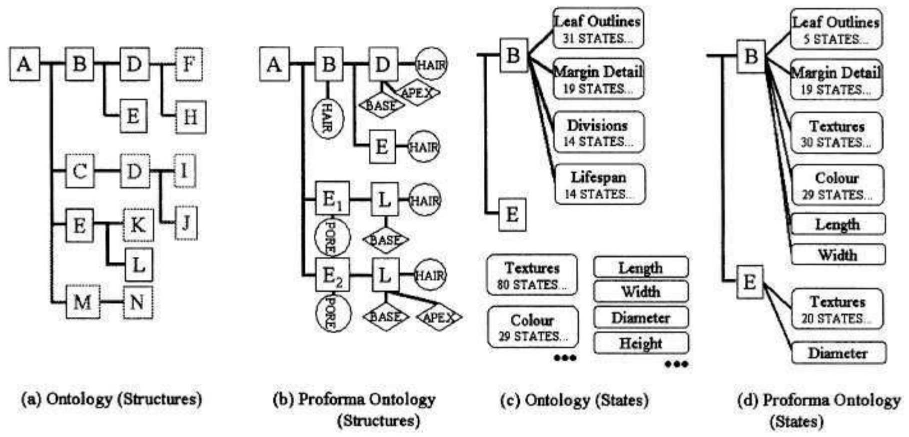


Fig. 6. (a) An ontology specifies all potential *Part-Of* paths in the structure hierarchy. (b) Many structures and paths may be deleted in an instance of a proforma ontology, some structures may be duplicated (*e.g. E*), and appropriate generic structures and regions explicitly included (*hairs, bases etc*). (c) Some qualitative state groups in the ontology are restricted to specific structures (*Divisions, Lifespan etc*), others (*Colours etc.*) and all quantitative properties are applicable anywhere. (d) Proforma ontologies may remove some allowed state groups for a structure, and possibly remove some allowed states from these groups; quantitative properties and unrestricted state groups will be explicitly attached to structures where desired for description.

3.4 Expanding the Ontology

At this stage we do not claim to have produced a complete ontology for the description of angiosperms, but have created an expandable ontology which can be augmented with the addition of more state and structure terms as required, providing that the addition of these terms does not alter the meaning of existing terms nor the interpretation of proformas or descriptions composed with earlier versions of the ontology. To this end new structures cannot be added within an existing structure path in the hierarchy, but a completely new path must be added (e.g. to insert a new structure *X* into the hierarchy [*C part-of B part-of A*: path A.B.C], so that [*X is a part-of B*] and [*C is a part-of X*], we retain the path A.B.C and introduce a new path [A.B.X.C] so that C now has two possible contexts, the original one still being valid). Similarly, new and existing state terms might be added to new or existing state groups, or new links between state groups and structures expressed.

It is possible to argue that the addition of new state terms to an existing set of states could alter the contextual meaning of all the states in that group. However, in order to maintain the compatibility of descriptions recorded with old and new versions of the group it is important to declare that the definitions of the terms are not altered by this process, and that the meaning is unambiguously captured in the textual definition of that state.

4 A Prototype Ontology and Future Work

A Java tool was developed which allowed the taxonomists to create and edit a prototype angiosperm ontology by entering defined terms and creating the various relationships shown in Figure 4. The ontology, stored in a relational database, contains over 1000 defined terms (term + definiton + citation). There are 24 Regions, 46 Generic Structures and 269 Structures of which 126 are defined as Types. 160 optional Part-Of relationships organize the 143 remaining Structures into a structural hierarchy, with only 19 Structure Terms currently described as potentially Part-Of more than one superstructure. The State Terms are distributed between 72 State Groups that reflect their usage context, with between 2 and 79 members of each group. 38 State Terms are members of more than one group (typically 2). A print out of the structure hierarchy represented as an expanded tree, and the whole ontology in XML format can be viewed [15]. Each of the 536 structure nodes in the tree is identifiable by its path; of these 331 are leaf nodes.

The path of each structure (node) in the structure tree is programmatically calculated and stored in the database. This path represents the identity of each node when included in description elements. We are currently exploring the most efficient way to store this in the database, as an adjacency table of node ID versus parent node ID; as a programmatically parseable string representation of the path (e.g. 'ID1.ID2.ID3'; see Figure 5c); or as an XML fragment representing the path as nested structure terms.

An additional tool is under development that will generate Data Entry Interfaces automatically using the input domain ontology, which can be then be specialized to create project-specific proforma ontologies as described in Section §3.3. This will allow the taxonomists to specify which structures and properties that they wish to describe for a given set of specimens, and create an electronic 'proforma' for data input. Specimen descriptions composed with this interface will be saved to a relational database compliant with our Description Element data model, using terms which are unambiguously defined in our angiosperm ontology. We propose to evaluate these tools and the validity and value of our character data model by using the system to capture real specimen descriptions using our angiosperm ontology. We hope to investigate to what extent it is possible to expand our ontology to describe wider plant taxa, or whether the creation of new ontologies will be necessary to describe disparate plant groups.

5 Discussion

It has been suggested that by committing to a publicly available ontology different data sources can ensure shared meaning and compatibility [16]. However, even if two systems share the same vocabulary there is no guarantee that their data can be integrated unless the ontologies commit to the same underlying conceptualization [17]. The Prometheus conceptual model defines how 'characters' can be represented in a common format, thus allowing description data to be shared between conformant sources and possibly with data sources with schemas that can be mapped to this model. An important distinction between the Prometheus description model, and other electronic description formats for taxonomy, is that the Prometheus methodology does not require that ‘characters’ are defined before description, but that actual observations can be recorded at description time using an ontology of defined terms.

Ontologies can be used to mediate both structural and semantic data integration by representing a unified view of local ontologies, or by sharing a common ontology amongst data providers [13,14,18]. Here we propose that the consistent representation of characters according to our flexible model will ensure structural and syntactic homogeneity. We propose that the more problematic issue of semantic heterogeneity (including problems of synonymy and homonymy) can be resolved by the use of a single common controlled vocabulary specified as an ontology. Indeed all descriptions created using our common parent description ontology will be compatible. However, if different taxonomic domains require distinct description ontologies, descriptions composed using separate ontologies will not be automatically compatible without an expert mapping of the concepts between ontologies, possibly by mapping to a generic integration ontology. Such mappings are often problematic and inexact, and fail to resolve all semantic conflicts that can result in data loss [13]. The ability to share information with legacy data collected without a well-defined terminology will be severely limited. For these reasons the creation and adoption of description ontologies with as wide a taxonomic range as possible is desirable. However, the current individualistic working practices of taxonomists make acceptance and adoption of an ‘imposed’ standardized description ontology unlikely. Rather, we hope that by creating and successfully demonstrating the use and benefits of an ontology in one taxonomic domain we will encourage the adoption and bottom up development and expansion of the ontology.

Detailed botanical ontologies are being developed by other groups, particularly the Plant Ontology Consortium (POC) [19-21]. POC are developing highly detailed anatomical and ‘trait’ ontologies, initially for three scientifically well-characterized model species (rice, maize and *Arabidopsis*). In many respects the level of detail specified in these ontologies goes beyond that required for taxonomic description, and being species-specific the ontologies are inappropriate for taxonomy.

There is a similar representation of structures according to POC’s anatomical ontologies and the Prometheus Ontology, with POC also recognizing the importance of ‘defined terms’ and relating these hierarchically using a central ‘Part-Of’ relationship. The POC ontologies, however, also incorporate an ‘Is A’ relationship, which is somewhat analogous to our Type attribute/relationship for structures, but which can fully participate in ‘Part-Of’ hierarchies. We found that incorporation of a full ‘Type Of’ relationship into our structural hierarchy made the ontology overly complex, particularly to non-experts, nor is it easy to agree meaningful ‘Type Of’ relationships across a large taxonomic range. POC ontologies include an additional ‘Derived From’ relationship, which expresses developmental information currently not represented in Prometheus.

The POC trait ontologies define genetically-based traits, mutations, phenotypes *etc.* rather than taxonomic ‘characters’. Furthermore, there is explicit linkage of traits to the Gene Ontology [20,22], which is inappropriate for the taxonomic domain, where there is typically virtually no genetic information available for specimens.

POC have adopted the Gene Ontology’s ‘True Path Rule’ which asserts that child terms in the relationship hierarchy inherit the meaning of all of their parent terms, thus the definitions of all parent terms on the path of a term must apply to that term [22]. Within Prometheus the hierarchical path of a given structure also has critical importance in determining its absolute structural context. However, the relationships expressed in the Prometheus ontology are only *possible* contexts, an actual structural context is only asserted when a term is used in a description. This distinction allows a flexible ontology that can be used across a wide taxonomic range.

We believe that the specialization of our parent description ontology into individual proforma sub-ontologies is a novel means for facilitating the collection of compatible description data. We also believe that capturing the rich semantic content expressed in our ontology, for example the ontologically defined context of a structure via its path, allows not only efficient and consistent knowledge sharing and reuse but will also allow rigorous representation and analysis of taxonomic concepts.

Development of our novel description methodology and data model can only be validated by providing tools to create, explore and use defined ontologies for specimen description, allowing taxonomists to record descriptions compliant with this constrained format. We have created an angiosperm ontology for the description of one taxonomic dataset and are extending it for the description of further test datasets. Providing tools that allow data entry using only a controlled defined terminology enforces semantic homogeneity, and will aid future integration of any database created using the tools.

References

1. Wilkinson, M.: A comparison of two methods of character construction. Cladistics 11 (1995) 297–308
2. Cannon, A., McDonald, S. M.: Prometheus II – Qualitative Research Case Study: Capturing and relating character concepts in plant taxonomy (2001) URL: www.prometheusdb.org/resources.html
3. Colless, D. H.: On 'character' and related terms. Systematic Zoology 34 (1985) 229-233 Keogh, J.S.: The importance of systematics in understanding the biodiversity crisis: the role of biological educators. Journal of Biol. Educ. 29 (1995) 293 – 299
4. Diederich, J., Fortuner, R., Milton, J.: Construction and integration of large character sets for nematode morpho-anatomical data. Fundamental and Applied Nematology 20 (1997) 409-424
5. DELTA: Dallwitz, M.J.: A general system for coding taxonomic descriptions. Taxon 29 (1980) 41-46
6. NEXUS: Maddison, D.R., Swofford, D.L., Maddison, W.P.: NEXUS: An extensible file format for systematic information. Systematic Biology 46 (1997) 590-621
7. LUCID: Developed by Centre for Biological Information Technology: University of Queensland, Australia. URL: www.cpitt.uq.edu.au, www.lucidcentral.com
8. Davis, P.H., Heywood, V.H.: Principles of Angiosperm Taxonomy. Oliver and Boyd Edinburgh. (1963)
9. TDWG (International Working Group on Taxonomic Databases) URL: www.tdwg.org; Structure of Descriptive Data.: Subgroup Session Report at the TDWG Meeting in Frankfurt (2000) www.tdwg.org/tdwg2000/sddreport.
10. Prometheus URL: www.prometheusdb.org
11. Allkin, R.: Handling Taxonomic Descriptions by Computer. In: Allkin R., Bisby F.A. (eds.): Databases in Systematics. Academic Press London (1984)
12. The Science Environment for Ecological Knowledge: URL: seek.ecoinformatics.org
13. Cui, Z., Jones, D.M., O'Brien, P.: Semantic B2B Integration: Issues in Ontology-based Applications. SIGMOD Record 31 (2002) 43-48

14. Omelayenko, B.: Syntactic-Level Ontology Integration Rules for E-commerce. In: Proceedings of the Fourteenth International FLAIRS Conference (FLAIRS-2001), Key West, FL. (2001) 324-328
15. URL: www.prometheusdb.org/resources.htm
16. W3C: OWL Web Ontology Language Use Cases and Requirements. URL: http://www.w3.org/TR/webont-req/ (2003)
17. Guarino, N.: Formal Ontology and Information Systems. In: Formal Ontologies in Information Systems. Proceedings of FOIS'98, Trento, Italy. IOS Press, Amsterdam (1998) 3-15
18. Sheth, A.: Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. In: Interoperating Geographic Information Systems. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, and C. A. Kottman (eds.), Kluwer, Academic Publishers, 1998, pp. 5-30.
19. Jaiswal, P., Ware, D., Ni, J., Chang, K., Zhao, W., Schmidt, S., Pan, X., Clark, K., Teytelman, L., Cartinhour,S., Stein, L., McCouch, S.: Conference Review: Gramene: Development and Integration of Trait and Gene Ontologies for Rice. Comparative and Functional Genomics 3 (2002) 132-136
20. The Plant Ontology Consortium: Conference Review: The Plant Ontology Consortium and Plant Ontologies. Comparative and Functional Genomics 3 (2002) 137-142
21. The Plant Ontology Consortium: URL: www.plantontology.org
22. The Gene Ontology Consortium: URL: www.geneontology.org

On the Application of Formal Principles to Life Science Data: a Case Study in the Gene Ontology

Barry Smith,[1,2] Jacob Köhler[3], and Anand Kumar[2]

[1] Department of Philosophy, University at Buffalo
[2] Institute for Formal Ontology and Medical Information Science, University at Leipzig
[3] Department of Bioinformatics, University of Bielefeld

Abstract. Formal principles governing best practices in classification and definition have for too long been neglected in the construction of biomedical ontologies, in ways which have important negative consequences for data integration and ontology alignment. We argue that the use of such principles in ontology construction can serve as a valuable tool in error-detection and also in supporting reliable manual curation. We argue also that such principles are a prerequisite for the successful application of advanced data integration techniques such as ontology-based multi-database querying, automated ontology alignment and ontology-based text-mining. These theses are illustrated by means of a case study of the Gene Ontology, a project of increasing importance within the field of biomedical data integration.

1 Introduction

In order to integrate databases, one has to overcome problems at several levels. These include *legal* problems related to access and redistribution, *technical* problems related to the employment of heterogeneous storage and access methods and query languages, and *housekeeping* problems relating to the management of corrections and updates. Here we are concerned with what are loosely called *semantic* obstacles to database integration [1], problems which manifest themselves both at the database schema level and at the level of data entry [2]. In simple terms, database systems should use language consistently, which means: they should use the same terms to refer to the same things. Semantic heterogeneities at the schema level arise where different names are used for equivalent database fields and tables. Systems to overcome schema level semantic heterogeneities are introduced in [3-8]. Semantic heterogeneities arise at the entry level where different terms are used for the same things, or the same terms are used for different things.

Ontologies have thus far played a role in the semantic integration of databases at the entry level by providing controlled vocabularies which have the goal of making it possible to search different databases secure in the knowledge that the same terms, wherever they are encountered, will always represent the same entities. One of the most impressive and influential developments in this respect is the Gene Ontology™ (GO) [9], which plays a central role in attempts to develop controlled vocabularies for shared use across different biological domains. The tremendous investment of time and effort by the GO Consortium has already brought considerable benefits to a range of different types of biological and biomedical research.

GO's December 2003 release contains some 16,658 terms divided into three networks whose topmost nodes are, respectively: *cellular component, molecular function* and *biological process.* In contradistinction to much existing usage, we shall refer to the nodes in such hierarchies not as *concepts* but rather on the one hand as *terms* (where we are interested in the hierarchies themselves, as syntactic structures), and on the other hand as *classes* (where we are interested in the biological entities to which these terms refer). It is classes, not concepts, which stand in *is a* and *part of* relations [10]. They do this in virtue of specific kinds of relations between the individual entities which instantiate these classes [10].

GO's three networks are structured by the relations of subsumption *(is a)* and of partonomic inclusion (*part of*). Crucially, GO treats its three structured networks as separate ontologies, which means that no ontological relations are defined between them. Thus for example GO does not record the fact that a given function is the *function of* a given component, or that a given process is the *exercise of* a given function. One reason for this is that the molecular functions and biological processes are standardly exercised by entities instantiating classes which fall outside the scope of GO. GO has an ontology of cellular components, not however of molecular structures or (with some few exceptions, such as *host* [GO:0018995]) of organism-level structures. Thus GO includes *behavior* [GO:0007610] and *adult behavior* [GO:0030534], defined as 'Behavior in a fully developed and mature organism', within its biological process hierarchies, but it has no room for *adult* or *organism.*

The decision of the GO Consortium to develop its three separate ontologies in this way has brought a variety of benefits, and we do not here recommend its abandonment. What we do recommend is that, in the process of constructing GO and of similar biomedical information resources in the future, special attention be paid to the problems which inevitably arise where standard rules of good classification are not respected – rules which have been tested in ontology research and in associated formal disciplines since the time of Aristotle and some examples of which are outlined below.

In the case of GO, these problems manifest themselves in characteristic shortfalls in formal integrity, turning on uncertainty as to the relations between the classes in GO's three ontologies. For example, there is no linkage between the term *taste* [GO:0007607], which is a biological process term, and the term *taste receptor activity* [GO:0008527], which is a molecular function term. This has an adverse impact on data integration when GO's data structure is used to retrieve or link related information from different data sources [11].

As we shall see, GO's three ontologies are not primarily used to support automated reasoning, but rather as a means by which biologists can easily and rapidly locate existing terms and relations within a structured vocabulary and so determine the proper treatment of new terms when the latter present themselves as candidates for incorporation in biomedical databases. To this end, GO has been highly successful in providing a controlled vocabulary that is used by different research groups in annotating genes and gene products. If, however, GO is to reap its full potential in supporting data integration across the life sciences, then shortfalls from formal integrity of the sorts here isolated become important. Here we shall focus on isolating such shortfalls and on suggestions as how they might be avoided in the future. Such avoidance is, we shall argue, a prerequisite for the use of GO in conjunction with tools for advanced data integration of a sort which meet the standards of contemporary biomedical informatics, including tools for alignment of ontologies [12, 13], ontology based text-mining [14-16] and 'intelligent' multi-database querying.

GO has often been criticized for its inconsistencies and for its lack of clear guidance as to what the relations between its three ontologies are. However, the literature thus far has not provided practical solutions to these problems in terms of general rules as to how they can be avoided in the future. Such problems cannot generally be overcome in an automated way. Rather, they require that the process of design and manual curation follow the general rules of classification which are familiar from the literature of logic and philosophy. Certainly biologists can avoid some inconsistencies via the use of ontology editing tools such as Protégé-2000, which support some automated consistency checking. However, by using Protégé-2000 the authors of [17] were able to identify only some minor inconsistencies in GO. It is, however, possible to take advantage of the fact that the Gene Ontology, like most of the controlled vocabularies united under the Open Biological Ontologies (OBO) umbrella, is maintained using the relatively user-friendly tool DAG-Edit [13]. One message of what follows is that one step towards greater formal coherence can be achieved by building sound policy into DAG-Edit and similar tools in ways which ensure the avoidance of certain kinds of errors, and we are pleased to see that OBO (http://obo.sourceforge.net/list.shtml) has defined relationships in development.

2 Problems with *Part of*

GO seeks to establish a 'controlled vocabulary'. This means that it accepts the rule of *univocity*: terms should have the same meanings (and thus point to the same referents) on every occasion of use. Unfortunately GO breaks this rule already in its treatment of the *part of* relation, which is at the very center of its hierarchical organization and with which at least three different meanings are associated [18]:

P1. *A part of B* means: *A is sometimes part of B* in the sense that there is for each instance of *A* some time at which it is part of an instance of *B* (in the standard mereological sense of 'part' as a relation between particulars). Example: *replication fork* (is at some times during the cell cycle) *part of nucleoplasm.*

P2. *A part of B* means: *A can be part of B* in the sense of a time-independent inclusion relation between classes, which we can summarize as: class *A* is part of class *B* if and only if there is some sub-class *C* of *B* which is such that all instances of *A* are included as parts in instances of *C* and all instances of *C* have parts which are instances of *A*. *Example*: *flagellum part of cell* (some types of cells have flagella as parts).

P3. *A part of B* means: vocabulary *A* is included within vocabulary *B*. Example: *cellular component ontology part of gene ontology.*

GO's 'part of ' violates not only *univocity* but also two other rules at the heart of good practice in the establishment of a formally rigorous framework of definitions: a term with an established use (inside and outside biomedical ontology) is used with a new, non-standard use; a lexically simple term is used to represent a lexically complex concept that is standardly expressed by means of phrases including the lexically simple term as part. To define 'part of' as meaning 'can be part of' is rather like defining 'sulphur' as meaning 'organo-sulphur compound.'

Solution: At present, DAG-Edit is shipped with only one built-in relation type, namely *'is a',* and even the latter can be deleted by the user at will. By including a fixed set of well-defined relation types that cannot be removed or modified by the user, biologists using DAG-Edit would become aware of the different relation types at their disposal. Whenever a user employs a relation type such as *'part of ',* a menu should pop up which offers a list of alternative more specific relation types, such as *is localized in* or *is involved in.* This would go far towards solving the problems which currently arise when OBO ontologies built with DAG-Edit use different names and identifiers for the same relation types and associate different relation types with the same names and identifiers.

3 Problems with *Is a*

GO's three term hierarchies have some obvious relation to the species and genera of more traditional biology When we evaluate GO as a classification of biological phenomena, however, then we discover that GO often uses *is a,* too, in ways which violate *univocity.* This is in part because GO is confined to the two relations *is a* and *part of* and because it does not allow ontological relations between its three separate ontologies. Thus it is constrained to place too great a load on *is a* than this relation is capable of bearing.

Thus when GO postulates:

[1] cell differentiation *is a* cellular process

[2] cell differentiation *is a* development,

then it means two different things by *'is a*', and only in the former case do we have to deal with a true subsumption relation between biological classes. That there is a problem with the latter case can be seen by noting that, where the agent or subject of differentiation is the *cell,* the agent or subject of development is the whole organism. That one process or function class subsumes another process or function class, implies, however, that same subject or agent is involved in each. This implies, however, that the definition:

> GO:0007275 **Development**
> **Definition:** *Biological processes specifically aimed at* the progression of an organism over time from an initial condition (e.g. a zygote, or a young adult) to a later condition (e.g. a multicellular animal or an aged adult)

should be modified by deleting the italicized portion, and that the relation in [2] should then more properly be expressed as: *contributes to.* Or alternatively, some way must be found to ensure that 'development' is used consistently in all its occurrences in GO, to refer either always to processes whose bearers are the whole organism, or to processes whose bearers may be individual cells, organs, or other organism constituents.

Another example of uncertainty regarding GO's *is a* relation is illustrated by:

[3] hexose biosynthesis *is a* monosaccharide biosynthesis

[4] hexose biosynthesis *is a* hexose metabolism,

Here the second *is a* seems more properly to amount to a *part of* relation, since *hexose biosynthesis* is just that part of hexose metabolism in which hexose is synthesized.

GO postulates:

[5] vacuole (sensu Fungi) *is a* storage vacuole

[6] vacuole (sensu Fungi) *is a* lytic vacuole,

where 'sensu' is the operator introduced by GO 'to cope with those cases where a word or phrase has different meanings when applied to different organisms,' (http://www.geneontology.org/doc/GO.usage.html#sensu).

Lytic vacuole is defined by GO as meaning: a vacuole that is maintained at an acidic pH and which contains degradative enzymes, including a wide variety of acid hydrolases. Inspection now reveals that *'is a',* here, stands in neither case for a genuine subsumption relation between biological classes. Rather, it signifies on the one hand the assignment of a *function* or *role,* and on the other hand the assignment of special features to the entities in question. Certainly there are, in some sense of the term 'class', classes of storage vacuoles and of lytic vacuoles; and certainly it is the case that all instances of *vacuole* (*sensu Fungi*) are instances of *storage vacuole* and of *lytic vacuole.* But that such relations obtain is not as yet sufficient for the existence of a genuine *is a* relation. A box used for storage is not (*ipso facto*) a special *kind* of box; rather it is a box which plays a special role in certain contexts. And similarly 'animal belonging to the emperor' does not designate a special *kind* of animal.

In Figure 1,

GO:0000327: **lytic vacuole within protein storage vacuole**

is recorded as standing in an *is a* relation to *protein storage vacuole.* In fact, however, we have to deal here with one of several ways in which GO expresses relations of *location.*

Another such example is GO's term 'unlocalized,' which is, and was always intended to be, a temporary term, but the consideration of which is nonetheless illuminating:

GO:0005941: **unlocalized**
Definition: Used as a holding place for cellular components whose precise localization is, as yet, unknown, or has not been determined by GO (the latter is the major reason for nodes to have this parent); this term should not be used for annotation of gene products,

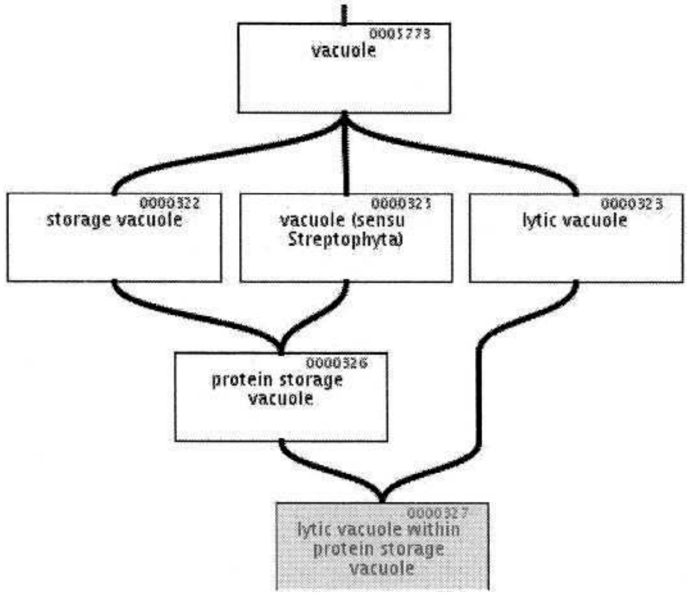


Figure 1: Treatment of *Vacuole* in GO
(taken from the QuickGO browser: http://www.ebi.ac.uk/ego)

is used for example in statements such as:

[7] Holliday junction helicase complex *is a* unlocalized

[8] Unlocalized *is a* cellular component

[7] and [8] illustrate GO's shortfall from two further principles:

positivity: complements of classes are not themselves classes. (Terms such as 'non-mammal' or 'non-membrane' do not designate natural kinds.)

objectivity: which classes exist is not a function of the current state of our biological knowledge. (Terms incorporating expressions such as 'unclassified' or 'with unknown ligand' do not designate biological natural kinds.)

Shortfalls from *positivity* are illustrated also by terms like 'non-muscle myosin'; shortfalls from *objectivity* by terms like 'G-protein coupled receptor activity, unknown ligand.'

Another rule of good classification familiar from the logico-philosophical literature is:

levels: the terms in a classificatory hierarchy should be divided into pre-determined levels (analogous to the levels of kingdom, phylum, class, order, etc., in traditional biology).

(Similarly, terms in a *partonomic* hierarchy should be divided into predetermined granularity levels, for example: organism, organ, cell, molecule, etc. GO's distinction between *biological* processes, *cellular* components and *molecular* functions/activities reflects this partonomic principle of levels, and GO currently contemplates a move to distinguish 'cell physiological process' and 'organism physiological process' in this same spirit.)

Theorists of classification have recognized further that the terms on each such level should ideally satisfy the rules:

> *single inheritance:* no class in a classificatory hierarchy should have more than one parent on the immediate higher level
>
> *exhaustiveness*: the classes on any given level should exhaust the domain of the classificatory hierarchy

These two rules together constitute the so-called JEPD (for: jointly exhaustive and pairwise disjoint) criterion. *Exhaustiveness* is of course often difficult to satisfy in the realm of biological phenomena. Shortfalls from *single inheritance* are however easy to detect, and their acceptance thus amounts to the rejection of the JEPD ideal.

The acceptance of the latter in traditional classifications is no accident. Exhaustiveness is a clear positive trait for a classificatory hierarchy; its acceptance in some form is presupposed as a goal by every biological scientist. Single inheritance reflects the presumption that if a term in a classificatory hierarchy has two *is a* parents, then the hierarchy needs to be refined [19]. Nowadays, however, single inheritance is less commonly accepted as a positive trait because multiple inheritance is so useful a device in facilitating compactness and efficient maintenance of large-scale ontologies. This is because it allows one to make large changes to a portion of an ontology without the need to adjust each individual lower-level element. It also allows one to avoid certain kinds of combinatory explosion. On the other hand, as will become clear from many of the examples treated in this paper, shortfalls from *single inheritance* are often clues to bad coding. This is because such shortfalls mark often subtle deviations from a reading of *'is a'* as standing for a genuine class subsumption relation. Such deviations are difficult to communicate to curators in terms of generally intelligible principles. But more importantly, they also serve as obstacles too ontology integration, since they amount to the conscious adoption of a policy according to which *'is a'* means different things in different contexts. [11]

We here leave open the question whether division into levels and single inheritance involving genuine *is a* relations can be achieved throughout the realm of classifications treated of by GO. We note only that, as Guarino and Welty [19] have shown, methods exist for systematically removing cases of multiple inheritance from class hierarchies by distinguishing *is a* relations from ontological relations of other sorts. Using these methods, well-structured classifications can be achieved by recognizing additional relations (for example: *has role, is dependent on, is involved in, contributes to, is located in*) and by allowing categories of entities of different sorts (for instance *sites, constituents, roles, functions, qualities*) within a single ontology. GO, however, does not have these alternatives at its disposal, not least because of its insistence that its three constituent vocabularies represent separate ontologies with no relations defined between them. At the same time, however, we will still need to find ways to represent in a formally coherent way those cases, such as *storage vacuole* or *immunological synapse,* where role and bearer together are referred to as forming a single entity.

4 Problems with Definitions

Like other biomedical ontologies, GO provides not only controlled vocabularies with hierarchical structures but also definitions of its terms. Indeed part of the goal of GO is to provide a source of strict definitions that can be communicated across people and applications. The definitions actually supplied by the GO Consortium, however, are affected by a number of characteristic problems which, while perhaps not affecting their usability by human biologists, raise severe obstacles at the point where the sort of formal rigor needed by computer applications is an issue, to the degree that much of the information content contained in GO's definitions is not accessible to software.

A well-structured definition should satisfy at least the rule:

> *intelligibility*: the terms used in a definition should be simpler (more intelligible, more logically or ontologically basic) than the term to be defined

– for otherwise the definition would provide no assistance to the understanding, and thus make no contribution to the GO Consortium's goal of formulating definitions which are usable by human beings.

That GO does not respect this rule is illustrated for example by:

> GO:0007512: **adult heart development**
> **Definition:** Generation and development of the heart of a fully developed and mature organism
>
> GO:0017145: **stem cell renewal**
> **Definition:** The self-renewing division of a stem cell. A stem cell is an undifferentiated cell, in the embryo or adult, that can undergo unlimited division and give rise to one or several different cell types.

Illustrates two features of GO's shortfall from standards of good practice in the formulation of its definitions: 1. the failure to draw a clear line between providing *definitions* of terms and *communicating extra knowledge,* 2. the inclusion within the definition of a GO of the definition of some second term, that is not present in GO. The solution to the latter problem is to provide a supplementary list of all non-GO terms treated in this fashion, together with their definitions. It would then be an interesting exercise to see what sort of ontology the terms included in this list would define.

Once again, the failure to follow a basic rule of classification and definition is a good clue as to the presence of coding errors. Thus consider:

GO:0016326: **kinesin motor activity**
Definition: The hydrolysis of ATP (and GTP) that drives the microtubular motor along microtubules,

The problem here is that hydrolysis is merely that which provides the energy for motor activity and not this activity as a whole. (Note that this term has been declared obsolete by GO, not however because it represents an error, but because it represents a gene product, and the latter do not fall within the scope of GO.)

GO:0008527: **taste receptor activity**
Definition: Combining with soluble compounds to initiate a change in cell activity. These receptors are responsible for the sense of taste.

fails to satisfy a further standard principle of good practice in definition, namely that of

substitutivity: in all so-called extensional contexts a defined term must be substitutable by its definition in such a way that the result is both grammatically correct and has the same truth-value as the sentence with which we begin.

(where an 'extensional context' is a context not within the scope of mental verb phrases such as 'I believe that', 'She denies that', 'He knows that'). Adhering to the rule of substitutivity ensures *inter alia* that definitions have the same grammatical form as the terms defined. Failures to adhere to this rule are particularly egregious in connection with GO's molecular function hierarchy.

It is not possible, on pain of infinite regress, to define every term in accordance with the principle of *intelligibility.* Rather, some terms must be left undefined. Definitions should more generally satisfy the rule of *modularity,* which means that they should be organized into levels, with level 0 terms being picked out as undefined primitives and terms on levels $n + 1$, for each $n \geq 0$ being defined by appeal exclusively to logical and ontological constants (such as 'and', 'all', 'is a' and 'part of') together with already defined terms taken from levels $\leq n$. Modular definitions are especially useful for the purposes of automatic reasoning.

Because the rules of *univocity, modularity* and *substitutivity* are not satisfied by GO's system of definitions, this means that substitution of the GO definitions for the corresponding defined terms appearing within other contexts can be achieved, at best, only with human intervention. A valuable source of automatic error-checking and of location of classificatory gaps is hereby sacrificed.

Consider the example:

GO:0003673: **cell fate commitment**
Definition: The commitment of cells to specific cell fates and their capacity to differentiate into particular kinds of cells.

The coarse logic of this definition is as follows:

x is a cell fate **commitment** $=_{\text{def}}$ x is a cell fate commitment and p, where *p* is, logically speaking, a second, extraneous condition. Here GO errs by providing in its definition at the same time too much and too little information, and the user does not know how to interpret the relation of the two clauses in the definition. (The first clause is marked, in addition, by the problem of circularity.)

If the rule of *modularity* is satisfied, then this means that in the case of compound terms the corresponding definitions should themselves be capable of being generated automatically. Thus for example the definition of 'garland cell differentiation' should be obtainable by taking the definition of 'cell differentiation' and substituting 'garland cell' for 'cell' throughout. What we find, however, is:

> GO:0030154: **cell differentiation**
> **Definition:** The process whereby relatively unspecialized cells, e.g. embryonic or regenerative cells, acquire specialized structural and/or functional features that characterize the cells, tissues, or organs of the mature organism or some other relatively stable phase of the organism's life history.
>
> GO:0007514: **garland cell differentiation**
> **Definition:** Development of garland cells, a small group of nephrocytes which take up waste materials from the hemolymph by endocytosis.

This leaves the user in a position where he does not know whether 'differentiation' as it occurs in the two contexts does or does not mean the same thing. GO's definition of 'garland cell differentiation' is marked further by the problem that it is in fact a definition of garland cell *development,* of which garland cell *differentiation* would in fact (given the definition of *development* provided earlier) be a proper subclass.

Solution: methods exist which can be used to control definitions for conformity with *modularity,* for example by highlighting words that occur in a definition even though they are located deeper in the *is a* hierarchy. Potential problems of circularity can also be indicated by highlighting non-logical words that occur in both a term and its definition. Compound terms can be recognized, and automatically generated definitions can be suggested to the user in terms of already existing definitions of the component parts.

5 Problems with 'Sensu'

A related set of problems can be illustrated by examining GO's use of its 'sensu' operator, which is introduced to cope with those cases where a word or phrase has different meanings when applied to different organisms, as for example in the case of *cell wall.* (Cell walls for example in bacteria and fungi have a completely different composition.) 'Using the *sensu* reference makes the node available to other species that use the same process/function/component' (http://www.geneontology.org/doc-/GO.usage.html#sensu). If, however, 'sensu' is designed to indicate that the modified term refers to a different class from that to which the unmodified term refers, then in what sense *are* we still dealing with 'the same process/function/component'?

Since the primary goal of the GO Consortium is to provide an ontology of gene products applying to all species, *sensu* terms are intended to be used sparingly. In consequence, *sensu* terms, as in the case illustrated in Figure 2, are allowed to have *non-sensu* terms as children, as in

[9] R7 differentiation *is a* eye photoreceptor differentiation (sensu Drosophilia).

For again, there is R7 differentiation in species other than Drosophilia, for example in crustaceans [20].

Another problematic example, which also illustrates once more GO's multiple ways of handling the relation of localization, is GO's postulation of:

[10] bud tip *is a* site of polarized growth (sensu Saccharomyces)

from which we can infer that:

[11] every instance of bud tip has an instance of Saccaromyces polarized growth located therein.

But GO also has:

[12] site of polarized growth (sensu Saccharomyces) *is a* site of polarized growth (sensu Fungi)

from which we can infer that:

[13] bud tip *is a* site of polarized growth (sensu Fungi)

and from there to:

[14] every instance of bud tip has an instance of Fungus polarized growth located therein.

[11] is consistent with [14] only if either (a) every instance of non-Saccharomyces Fungus polarized growth is co-localized with an instance of Saccharomyces polarized group or (b) there is Fungus polarized growth only in Saccharomyces. (a) we take to be biologically false; but (b) implies that 'site of polarized growth (sensu Saccharomyces)' and 'site of polarized growth (sensu Fungi)' in fact refer, confusingly, to the same class, and thus that the latter should be removed from GO's cellular component ontology.

A further problem is caused by GO's use of 'sensu Invertebrata'. Whereas *vertebrate* is a well-defined biological taxon, biologists tend to disagree on what the definition of *invertebrate* should be, and thus apply the 'sensu Invertebrata' modifier to different taxa. The resultant errors are illustrated for example in the genes annotated to

GO:0006960 **: antimicrobial humoral response (sensu Invertebrata)** in the browser AMIGO, many of which are not invertebrate genes but rather human genes (for example COPE HUMAN, PTGE HUMAN, PTE1 HUMAN, and so on). It is surely obvious that a gene with suffix 'HUMAN' should not be annotated to a biological process which is assigned to invertebrates.

Solution: These and other problems can be overcome by introducing 'sensu' as a relational expression with a well-defined meaning that references a systematic species nomenclature such as the TAXON database [21]. We understand that GO is contemplating taking steps along these lines. In addition measures can be taken to check automatically that all GO terms that occur in a given branch of an '*is a*' hierarchy use the same taxon. The problem with 'Invertebrata' is also overcome, in virtue of the fact that no standard systematic species nomenclature contains this term.

6 Problems with 'Function'

Recall GO's definition of 'toxin activity' as: 'Acts as to cause injury to other living organisms.' The problem here flows from GO's unstable view of how the classes in molecular function ontology should precisely be designated [22]. The same problem makes itself manifest also in cases such as:

> GO:0005199: **structural constituent of cell wall**
> **Definition:** The action of a molecule that contributes to the structural integrity of a cell wall,

where the definition confuses *constituents* (which ought properly to be included in GO's constituent ontology) with *activities,* which GO includes in its function ontology. Many other constituents are similarly defined as activities:

extracellular matrix structural constituent
puparial glue (sensu Diptera)
structural constituent of bone
structural constituent of chorion (sensu Insecta)
structural constituent of chromatin
structural constituent of cuticle
structural constituent of cytoskeleton
structural constituent of epidermis
structural constituent of eye lens
structural constituent of muscle
structural constituent of myelin sheath
structural constituent of nuclear pore
structural constituent of peritrophic membrane (sensu Insecta)
structural constituent of ribosome
structural constituent of tooth enamel
structural constituent of vitelline membrane (sensu Insecta)

7 An Alternative Regime of Definitions

As a brief illustration of a regime of definitions built up in such a way as to satisfy the principles listed above, we consider the Foundational Model of Anatomy (FMA), which is being developed at the University of Washington, Seattle as part of the Digital Anatomist Project. The term hierarchy of the FMA consists in a symbolic representation of the structural organization of the human body from the macromolecular to the macroscopic levels, with the goal of providing a robust and consistent scheme for classifying anatomical entities which can serve as a reference ontology in biomedical informatics [23].

FMA seeks to follow the formal rules for definitions laid down by Aristotle. A definition, on this account, is the specification of the essence (nature, invariant structure) shared by all the members of a class or natural kind. Definitions are specified by working through a classificatory hierarchy from the top down, with the relevant topmost node or nodes acting in every case as undefinable primitives. The definition of a class lower down in the hierarchy is then provided by specifying the parent of the class (which in a regime conforming to single inheritance is of course in every case unique) together with the relevant differentia, which tells us what marks out instances of the defined class or species within the wider parent class or genus, as in: *human* = *rational animal,* where *rational* is the differentia. An Aristotelian definition then satisfies the condition that an entity satisfies the defining condition if and only if it instantiates the corresponding class.

Thus definitions in FMA look like this:

> **Cell** ***is a*** **anatomical structure** that *consists of* **cytoplasm** *surrounded by* a **plasma membrane** with or without a **cell nucleus**
>
> **Plasma membrane** *is a* **cell part** that *surrounds* the **cytoplasm,**

where terms picked out in bold are nodes within the FMA classification and italicized terms signify the formal-ontological relations – including *is a* – which obtain between the corresponding classes.

As the FMA points out, ontologies ‘differ from dictionaries in both their nature and purpose’ [24]. Dictionaries are prepared for human beings; their merely nominal definitions can employ the unregimented resources of natural language, can tolerate circularities and all manner of idiosyncrasy. In ontologies designed to be usable by computers, however, definitions must be formally regimented to a much higher degree. The specific type of regimentation chosen by the FMA has the advantage that each definition reflects the position in the hierarchy to which a defined term belongs. Indeed the position of a term within the hierarchy enriches its own definition by incorporating automatically the definitions of all the terms above it. The resultant system of definitions brings the benefit that the entire information content of the FMA’s term hierarchy can be translated very cleanly into a computer representation, and brings also further benefits in terms of reliable curation, efficient error checking and information retrieval, and ease of alignment with neighboring ontologies. The FMA defines an ontology as a ‘true inheritance hierarchy’ and it thereby draws attention to the fact that one central reason for adopting the method of ontologies in supporting reasoning across large bodies of data is precisely the fact that this method allows the exploitation of the inheritance of properties along paths of *is a* relations. FMA's regime of definitions – unlike that of GO – gives due merit to this principle.

8 Conclusion

We are not proposing here that GO abandon all its current practices in structuring its ontologies and accompanying definitions. The world of biomedical research is clearly not concerned with all of those sorts of scrupulousness that are important in more formal disciplines. Rather, it is a world of difficult trade-offs, in which the benefits of formal (logical and ontological) rigor need to be balanced on the one hand against the constraints of computer tractability, and on the other hand against the needs of practicing biologists. All the formal rules presented above should therefore be conceived as rules of thumb, or as ideals to be borne in mind in practice, rather than as iron requirements.

Note, too, that we are not suggesting that the problems outlined in the above have led to concrete errors in the annotations of genes by third parties, for example in the construction of model organism databases. We hypothesize that the biologists who are responsible for such annotations are able to use their biological expertise in order to block the faulty inferences which would otherwise result. To the extent, however, that GO is pressed into service as a reference-platform for the *automatic* navigation between biomedical databases, then the issue of consistency with standard principles of classification and definition will come to be of increasing importance.

Some of the mentioned problems can be overcome via relatively minor modifications to DAG-Edit, which would have a significant impact on the design and reliability of GO's ontologies since they would sharpen the awareness of designers and users in ways which can lead both to the avoidance of common pitfalls in the course of curation and to an ontologically more coherent structuring of the resultant data. The advantage in incorporating these changes into DAG-Edit would be also that it would not require that GO and the other OBO ontologies be rebuilt from scratch: actual and potential inconsistencies would be highlighted, and can be corrected on the fly.

Multiple inheritance, to repeat, is a particularly important cause of problems in the guise of both coding errors and obstacles to the coherent alignment of ontologies that will be needed in the future. This is because the success of such alignment depends crucially on the degree to which the basic ontological relations – above all relations such as *is a* and *part of* – can be relied on as having the same meanings in the different ontologies to be aligned. Cases of multiple inheritance go hand in hand, at least in many cases, with the assignment to the is *a* relation of a plurality of different meanings within a single ontology. The resultant melange makes coherent integration across ontologies achievable (at best) only under the guidance of human beings with the sorts of biological knowledge which can override the mismatches which otherwise threaten to arise. This, however, is to defeat the very purpose of constructing bioinformatics ontologies as the basis for a new kind of biological and biomedical research designed to exploit the power of computers.

As Ogren *et al.* [25] have pointed out, almost two-thirds of all GO terms contain other GO terms as substrings, the including term being in many cases derived from the included term via operators such as ‘regulation of’ or ‘sensu’. Many of the latter recur consistently in certain kinds of subtrees of GO’s three ontologies, and in ways which reflect ontologically significant relations between the corresponding classes. Ogren *et al.* propose that the presence of these operators be exploited ‘to make the information in GO more computationally accessible, to construct a conceptually richer representation of the data encoded in the ontology, and to assist in the analysis of natural language texts.’ We suggest taking this proposal still further by building the corresponding machinery for enforcing compositionality into the DAG-Edit tool and by exploiting analogous compositionality of information on the side of GO’s definitions. Such proposals will, however, bear fruit only to the extent that GO’s classifications and definitions satisfy the formal principles set forth above.

Acknowledgements
This paper was written under the auspices of the Wolfgang Paul Program of the Alexander von Humboldt Foundation. Thanks are due also to Stuart Aitken, Bert R. E. Klagges, Steffen Schulze-Kremer, Jane Lomax, Cornelius Rosse and Jean-Luc Verschelde.

References

[1] Köhler, J.: Integration of Life Science Databases. BioSilico March (2004)
[2] Kim, W., Seo, J.: Classifying Schematic and Data Heterogeneity in Multidatabase Systems. IEEE COMPUTER 24 (1991) 12-18
[3] Madhavan, J., Bernstein, P. A., Rahm, E.: Generic Schema Matching with Cupid. In: Proc. 27th Int. Conf. on Very Large Data Bases (VLDB) (2001)
[4] Köhler, J., Philippi, S., Lange, M.: SEMEDA: Ontology Based Semantic Integration of Biological Databases. Bioinformatics 19 (2003)
[5] Köhler, J., Lange, M., Hofestädt, R., Schulze-Kremer, S.: Logical and Semantic Database Integration. In: Proc. Bioinformatics and Biomedical Engineering (2000) 77-80
[6] Baker, P. G., Brass, A., Bechhofer, S., Goble, C., Paton, N., Stevens, R.: TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. An Overview. In: Proc. sixth International Conference on Intelligent Systems for Molecular Biology (1998)
[7] Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N. W., Goble, C. A., Brass, A.: TAMBIS: transparent access to multiple bioinformatics information sources. Bioinformatics 16(2000) 184-5
[8] Ludäscher, B., Gupta, A., Martone, M. E.: Model-Based Mediation with Domain Maps. In: Proc. 17th Intl. Conference on Data Engineering (ICDE), (2001)
[9] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., Sherlock, G.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nat Genet 25 (2000) 25-9.
[10] Smith, B.: The Logic of Biological Classification and the Foundations of Biomedical Ontology (Invited paper). In: Proc. 10th International Conference in Logic Methodology and Philosophy of Science, Oviedo, Spain, 2003. (2003)
[11] Lord, P. W., Stevens, R. D., Brass, A., Goble, C. A.: Semantic similarity measures as tools for exploring the gene ontology. Pac Symp Biocomput (2003) 601-12
[12] Knight, K., Luk, S.: Building a Large-Scale Knowledge Base for Machine Translation. In: Proc. National Conference on Artificial Intelligence - AAAI (1994)

[13] Lambrix, P., Habbouche, M., Perez, M: Evaluation of ontology development tools for bioinformatics. Bioinformatics 19 (2003) 1564-71

[14] Kim, J. D., Ohta, T., Tateisi, Y., Tsujii, J.: GENIA corpus-a semantically annotated corpus for bio-textmining. Bioinformatics 19 Suppl 1 (2003) I180-I182

[15] Nenadic, G., Mima, H., Spasic, I., Ananiadou, S., Tsujii, J.: Terminology-driven literature mining and knowledge acquisition in biomedicine. Int J Med Inf 67 (2002) 33-48

[16] Chiang, J. H., Yu, H. C.: MeKE: discovering the functions of gene products from biomedical literature via sentence alignment. Bioinformatics 19 (2003) 1417-22

[17] Yeh, I., Karp, P. D., Noy, N. F., Altman, R. B.: Knowledge acquisition, consistency checking and concurrency control for Gene Ontology (GO). Bioinformatics 19 (2003) 241-8

[18] Smith, B., Rosse, C.: The Role of Foundational Relations in Biomedical Ontology Alignment. (under review)

[19] Guarino, N., Welty, C.: Identity and subsumption. In: R. Green, C. A. Bean, and S. Hyon Myaeng, (eds.): The Semantics of Relationships: An Interdisciplinary Perspective. Kluwer Academic Publishers (2002) 111–126

[20] Hafner, G. S., Tokarski, T. R.: Retinal development in the lobster Homarus americanus. Comparison with compound eyes of insects and other crustaceans. Cell Tissue Res 305 (2001) 147-58

[21] Wheeler, D. L., Church, D. M., Federhen, S., Lash, A. E., Madden, T. L., Pontius, J. U., Schuler, G. D., Schriml, L. M., Sequeira, E., Tatusova, T. A., Wagner, L.: Database resources of the National Center for Biotechnology. Nucleic Acids Res 31 (2003) 28-33

[22] Smith, B., Williams, J., Schulze-Kremer, S.: The Ontology of the Gene Ontology. In: Proc. Annual Symposium of the American Medical Informatics Association (2003) 609-613

[23] Rosse, C., Mejino Jr., J. L. V.: A Reference Ontology for Bioinformatics: The Foundational Model of Anatomy. Journal of Biomedical Informatics in press (2003)

[24] Michael, J., Mejino, J. L., Jr., Rosse, C.: The role of definitions in biomedical concept representation. Proc AMIA Symp (2001) 463-7

[25] Ogren, P. V., Cohen, K. B., Acquaah-Mensah, G. K., Eberlein, J., Hunter, L. T.: The Compositional Structure of Gene Ontology Terms. In: Proc. Proceedings of the Pacific Symposium on Biocomputing - PSB (2004)

Index-Driven XML Data Integration to Support Functional Genomics

Ela Hunt[1], Evangelos Pafilis[1,2], Inga Tulloch[1,2], and John Wilson[2]

[1] Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK
ela@dcs.gla.ac.uk

[2] Department of Computer and Information Sciences, University of Strathclyde, Glasgow G1 1XH, UK
jnw@cis.strath.ac.uk

Abstract. We identify a new type of data integration problem that arises in functional genomics research in the context of large-scale experiments involving arrays, 2-dimensional protein gels and mass-spectrometry. We explore the current practice of data analysis that involves repeated web queries iterating over long lists of gene or protein names. We postulate a new approach to solve this problem, applicable to data sets stored in XML format. We propose to discover data redundancies using an XML index we construct and to remove them from the results returned by the query. We combine XML indexing with queries carried out on top of relational tables. We believe our approach could support semi-automated data integration such as that required in the interpretation of large-scale biological experiments.

1 Introduction

The field of functional genomics promises to provide an understanding of the cellular processes that control development, health and disease. The advent of large-scale laboratory techniques including sequencing, microarrays and proteomics has revolutionised the discipline and allowed biologists to take a global view of biological processes. A sequencing run[3], a microarray experiment[4] or a 2-D protein gel[5] suddenly deliver a large amount of data that needs to be assessed with relationship to a particular problem being investigated. Current data acquisition and processing methods that rely on web browsing and querying for single genes of interest are no longer sufficient. A microarray experiment can yield a list of 500 gene names, all possibly implicated in the mechanism of a biological process or disease that is being investigated. Tools such as Sequence Retrieval System (SRS)[6] [10] provide a means of executing a query over a range of data sources but do not integrate the results. Keyword searching can be used to limit

[3] http://www.sanger.ac.uk/HGP/
[4] http://www.mged.org
[5] http://psidev.sourceforge.net/
[6] http://srs.ebi.ac.uk

the range of results but still produces large numbers of links that need to be investigated. The biologists we work with are able to identify lists of genes but have not solved the problem of data and literature acquisition that would help to place each gene in its context. The interpretation of gene expression and protein expression data requires data integration on a large scale. We believe that a new *generic* approach to data integration is needed. We can no longer afford to manually create new wrappers, mappings and views expressing the information need of every biologist performing genome or proteome analysis. A generic solution would be preferable. Since a significant number of biological databases are available in XML format, we have the opportunity to use that format for semi-automated data integration. We propose an XML data integration system that can gather all available XML data for any gene of interest and present the biologist with a digest of all known information, similar to GeneCards [16] but not limited to the *Homo sapiens* species.

The eXtensible Markup Language, XML[7], is one of the alternative formats used in biological data presentation. In particular, the SRS system handles XML easily and provides tools for the indexing of such data. SRS does not however integrate data, instead, integrated data views have to be crafted by skilled programmers. The biological users we know have to manually retrieve and integrate data. It is our intention to provide a data integration service for such users. We propose a different scenario. Our user is a biologist performing a large-scale biological experiment. The experiment produces a list of gene or protein names and our task is to find all available information about those data items from the public web databases.

We believe that a gene- or protein-focused view of all biological databases can satisfy the needs of a large number of researchers. An XML tree giving a summary of all information relevant to a given gene or protein could be processed by a data integration system and stored as a pre-processed summary. Each user would then select from that summary a relevant subset of information. Creating an XML tree for a gene can be done simplistically, by creating a root and attaching a gene-specific subtree from each database to that root. However, this solution is not satisfactory because the same data will be replicated in many subtrees. We aim to integrate the data at the XML level while taking into account both the data structure and content. We adopt ideas from the data mining community where both the tree structure and content are considered. We combine this idea with the use of database indexes that will make possible an approach similar to data mining on very large data sets. By indexing we reduce the computational complexity of the processing needed to find correspondences between different paths in XML trees, and by using simple queries on all indexed data we can develop well-founded mappings to be used to remove duplicated information.

Our work is based on the following assumptions and observations. We observe that biological databases are developed independently, and their XML structures have various levels of nesting and often use disjoint vocabularies in their tree structures. Because the data comes from various domains (for instance

[7] http://www.w3.org/TR/REC-xml

eukaryotic promoters, protein structures, genome maps), it is unlikely that automated matching of attribute names will allow us to merge the data. Each biologist has different needs and may use any number of databases. It is consequently impossible to prescribe how data should be merged and presented. On the other hand, we expect a certain degree of order and data sharing between different database trees. We assume shared database identifiers, unique IDs for single records (because most databases started as flat files each entry will have an ID) and large numbers of records of similar structure.

Our contributions are as follows. We identify a new type of data integration requirement. We provide a critique of existing data integration systems and we propose the idea of item-based XML data integration. We propose an index-based algorithm for redundancy removal and an application architecture to support querying. The rest of the paper follows this order and closes with a discussion and conclusions.

2 Searching On-line Databases

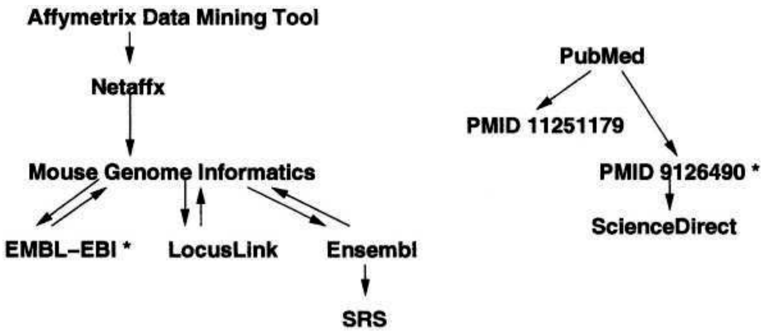


Fig. 1. Webpages visited during searching. * indicates sites that contained material that the researcher found interesting

Biologists have a variety of aims when searching on-line databases. We conducted extended interviews with four research groups that use microarray and proteomics technologies. We observed the process of data analysis performed as a series of web queries. We gathered logs of web activity using Muffin[8] and focused on web queries and data navigation paths.

2.1 Study 1, Mouse Breast Development as a Model for Cancer

Study 1 aims to understand gene expression in mouse mammary gland development and involution. A series of microarray experiments has led to the identifi-

[8] www.muffin.org

cation of 400 gene names. All genes should be studied and potential candidates identified on the basis of a known role in cancer and mammary gland development. The researcher reported spending six months characterising 100 genes by web browsing. He would like to be able to study the remaining 300 genes. The current practice is to start with the Affymetrix database[9] and for each probe name identified in the experiment, to find the corresponding gene. The link to the gene is followed, finally leading to PubMed articles[10]. If the article title correlates with the research interest of *Study 1,* it will be downloaded. The sequence of a typical search is shown in Figure 1. Leading on from the Affymetrix Data Mining Tool, the researcher found useful material in EMBL, but ultimately resorted to PubMed to issue a query for the gene name listed in the Affymetrix database. PubMed produced two articles that were of interest, one of which was pursued through to Science Direct[11], and ultimately printed out for subsequent study. This process was followed for all 100 genes and the researcher would like to be able to automate it. Moreover, he would like an additional facility, that of grouping the articles according to the number of the gene names in the query set they refer to. For instance, articles could be presented in a ranked order with those shown first that mention the greatest number of gene names out of the initial query set.

2.2 Study 2, Search for Candidate Genes for Hypertension

Study 2, working on a rat model of hypertension, wants to identify the exact genome location for 400 genes and to find supplementary information about similar genes in mouse and human. The web search using Ensembl[12], NCBI[13], MGD[14] and RGD[15] consumed a significant amount of time, simply to find gene locations. The identification of up-to-date comparative maps of human, mouse and rat for a subset of the genes located on one rat chromosome was also time-consuming. The search started from the Affymetrix database and the initial list of gene positions was gathered by web searching. The list was complemented with additional probe positions calculated by running a large-scale sequence comparison where probes from the Affymetrix database were compared to the rat genome.

2.3 Study 3, Rat Model of Schizophrenia

Study 3 is characterising 250 genes that are differentially expressed in a rat model of schizophrenia. We estimate that a full analysis of those genes will consume

[9] www.affymetrix.com
[10] www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed
[11] www.sciencedirect.com
[12] www.ensembl.org
[13] www.ncbi.nlm.nih.gov/LocusLink/
[14] www.informatics.jax.org
[15] rgd.mcw.edu

several months. The Affymetrix database of probe names is the starting point. Web queries on probe sequences lead the researcher to the DNA sequence of the rat gene or its human or mouse homologue. Those target sequences will then be used to design laboratory tests and relevant publications will be sought in PubMed and downloaded.

2.4 Study 4, *Trypanosoma brucei* Proteins

Study 4 is characterising the proteome of *Trypanosoma brucei* by 2-D gel electrophoresis [8] and mass spectrometry. Mass spectrometry-based protein identification is based on sequence searching. For each protein similar to the one observed on a 2-D gel, the study will want to gather information from the web, possibly extending to known publications, predicted genes, protein structures or protein motifs. Correlation with any published experimental data, including microarrays, would be beneficial to the study.

3 Existing Integrative Tools

3.1 Entrez

Entrez is possibly the most popular biological query system[16]. It offers a query interface which allows the biologist to distribute the query over all of Entrez databases. However, there is no further data integration. Query results list each of the underlying data sources together with the corresponding number of hits in each database. Clicking on any of the databases which show a match will deliver a list of links to data items stored in each database.

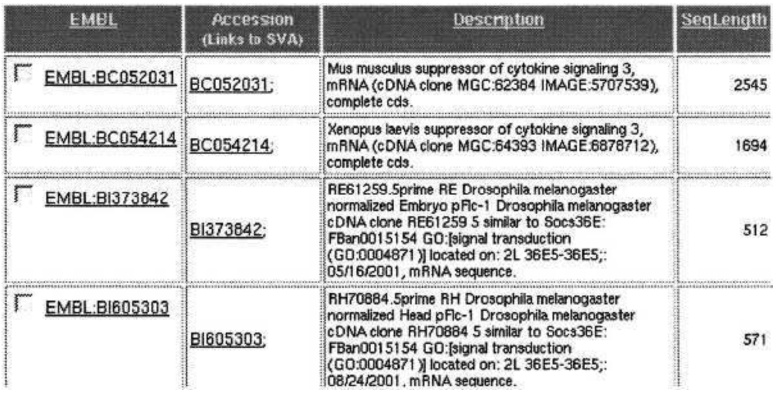
3.2 Sequence Retrieval System (SRS)

SRS is a warehousing and indexing system [10]. The system regularly downloads flat files and XML databases during low usage periods. Indexes are constructed in order to support queries over a number of data sources. Relational databases are not mirrored but their query facilities are captured in such a fashion that the databases can be queried effectively over the Internet. A user can select a set of target databases and issue a query spanning a selection of data sources. The query is then sent to remote relational databases and to the local flat files and XML data sources. The results are then brought together for the user. The results are presented as web pages of links or tables of data. Figure 2 shows an example result set.

3.3 GeneCards

The information content of GeneCards [16] focuses on the human genome. Data is integrated either by establishing local caches or by executing queries against

[16] www.ncbi.nlm.nih.gov/Entrez/



EMBL	Accession (Links to SVA)	Description	SeqLength
EMBL:BC052031	BC052031;	Mus musculus suppressor of cytokine signaling 3, mRNA (cDNA clone MGC:62384 IMAGE:5707539), complete cds.	2545
EMBL:BC054214	BC054214;	Xenopus laevis suppressor of cytokine signaling 3, mRNA (cDNA clone MGC:64393 IMAGE:6878712), complete cds.	1694
EMBL:BI373842	BI373842;	RE61259.5prime RE Drosophila melanogaster normalized Embryo pFlc-1 Drosophila melanogaster cDNA clone RE61259 5 similar to Socs36E: FBan0015154 GO:[signal transduction (GO:0004871)] located on: 2L 36E5-36E5;: 05/16/2001, mRNA sequence.	512
EMBL:BI605303	BI605303;	RH70884.5prime RH Drosophila melanogaster normalized Head pFlc-1 Drosophila melanogaster cDNA clone RH70884 5 similar to Socs36E: FBan0015154 GO:[signal transduction (GO:0004871)] located on: 2L 36E5-36E5;: 08/24/2001, mRNA sequence.	571

Fig. 2. SRS nucleotide database query result for *socs3*

on-line databases. Local caches are held in flat file format although this is currently being migrated to XML. Data is integrated from around 50 sources including Swiss-Prot, OMIM, Ensembl and LocusLink. The search engine removes redundancy and presents a digest of the information extracted. The HUGO database is used to identify the aliases of a particular gene. The results page presents a synopsis of the function of the gene and also a link to PubMed to allow the identification of literature that describes the gene. Source articles are not directly referenced on the results page but a variety of links (in addition to PubMed) are provided that lead to publications.

3.4 MedMiner

Medminer [17] is a text mining tool. It filters article abstracts and presents the most relevant portions. It consists of three components: the facility that queries web databases, the text filtering engine and the user interface. PubMed and GeneCards are used by MedMiner as information sources. The GeneCards query is used to return genes that are relevant to the user. The user can then select one of those genes for inclusion in a PubMed query. Arguments can be added to the PubMed query to initiate the filtering process. The abstracts of the returned articles are broken into sentences and each sentence is tested to see whether it conforms to the relevance criterion. The latter is true when the sentence contains the query arguments and one of the MedMiner keywords. These keywords are truncated terms associated with biological processes. When the user wishes to study the genes associated with a certain phenomenon, the co-occurrence of a gene identifier with a phenomenon relevant term in the abstract of an article increases the possibility that this article is an important one. After the text mining has been concluded, the citations that pass the relevance filter are grouped according to the keywords they contain. MedMiner, instead of showing the whole article abstract, displays only the sentence found to be relevant, with the keywords and the query argument, highlighted. The use of keywords as the article relevance metric runs the risk of high false negatives, especially if the keywords have not been wisely selected and they do not cover all the aspects of a certain phenomenon or process. A potential limitation is that relationships between keywords and gene identifiers that span multiple sentences will not be picked up. The choice of displaying only the relevant sentence instead of the abstract itself, can result in significant time savings. However there is a trade off with the possibility that important pieces of information existing in the rest of the abstract will never be displayed.

3.5 Limitations of Existing Tools

GeneCards is specific to the human genome and does not cover mouse, rat, or other organisms. MedMiner inherits the same weakness. The GeneCards system is based on hard-coded schema mappings and is not easily extended to add new data resources, or new XML files acquired from laboratory equipment or from collaborators. The approach is not flexible or scalable, however it represents a significant step in data integration methods. In fact, an approach that combines both literature and database resources and is extensible to any species of interest is required. Ideally, a PubMed abstract could be co-indexed with terms found in traditional databases to achieve an integration of textual, experimental and other annotation data present in public repositories. Further to that, queries over sequence data should be part of the data integration system allowing the integration of textual, database and sequence data to be even more effective.

4 Structural Conflicts in Data

The significance of XML in the integration of biological databases has been recognised for some time [1]. Some databases, such as Swiss-Prot, are available for download in XML format whereas other systems such as PubMed are able to return the results of queries as XML. The increasing conformance to the XML standard will help with the outstanding issues of integration between heterogeneous databases. Significant difficulties will however remain as a result of the current divergence in data design and the expected continuation of this divergence. Figures 3 and 4 show parts of the XML trees for the Swiss-Prot database and queries returned from PubMed. The gene name in Swiss-Prot is found by following the path: sptr/entry/reference/gene/name. There is no equivalent path in PubMed, but the same name may be found in an abstract, by following the path: PubMedArticleSet/PubMedArticle/MedlineCitation/Article/-Abstract/AbstractText. There is also another connection between the two trees. A Swiss-Prot entry records a PubMed article as a leaf on the path sptr/-entry/reference/citation/dbReference and the key used by Swiss-Prot is the key present in the PubMed tree reachable via PubMedArticleSet/PubMedArticle/-MedlineCitation/PMID. It can clearly be seen that the two structures contain

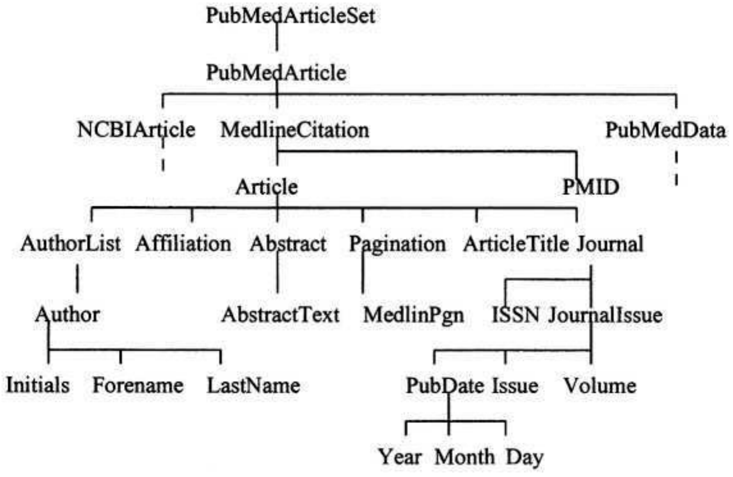


Fig. 3. Tree structure for PubMed query results

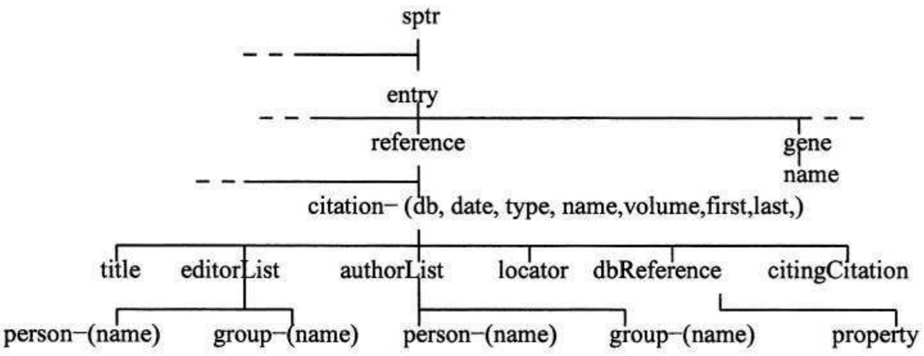


Fig. 4. Tree structure for the Swiss-Prot database. Attributes associated with tag names are bracketed

several incompatibilities. Although there are some matches between the structures at the level of tag names (eg *authorList*) there are many instances of data items that contain common semantics but are represented by varying structures. Dates in Swiss-Prot are represented by a single string giving the month and year whereas in PubMed they are represented by a sub-tree. Attributes in Swiss-Prot contain equivalent semantic content that is held by tags and their values in PubMed.

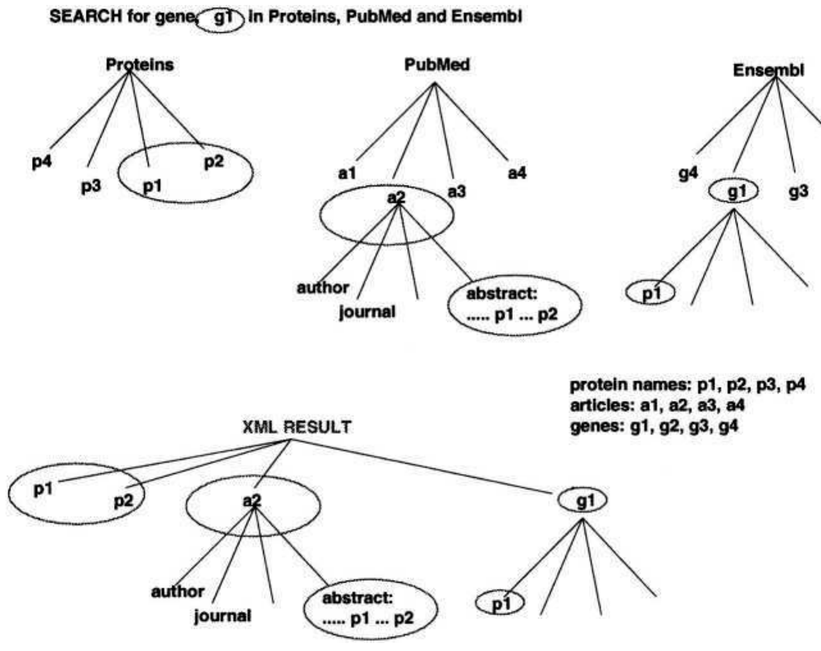


Fig. 5. A schematic view of data querying and integration using XML trees

5 XML Result Construction

The four biological studies we reported on are characterised by the use of a variety of databases that are manipulated using web interfaces, most of which offer a simple query modality and return the result as a mixture of textual information and web links. The starting point of the search is the result from a laboratory experiment. In the microarray context it is a list of probe names and in proteomics it is a list of peptide masses delivered by a mass spec instrument. In both cases the researchers enter their query in one of the available interfaces and follow a list of links returned by the query. The resulting data returned from each of those queries represents a subtree of an XML tree for each database being queried. Figure 5 shows a query for a gene *g1* which returns two items *p1, p2* from a protein database, an article *a2* from PubMed and a gene *g1* from Ensembl. Conceptually, all the information returned by those queries (or obtained by following the links between databases), represents a union of three XML subtrees. These subtrees can be brought together, by adding a root and attaching the subtrees to it. We believe that we could build such XML trees for each gene of interest. This approach would go further than the SRS system which presents the user with a list of items. One of the difficulties the user has with SRS is that the data returned from the queries is presented as a sequence of items, rather than integrated in such a way as to present a unified result.

This forces the researcher to follow the links, save the web pages and then deal with a large body of heterogenous data that contains duplication and is not ordered in a meaningful way. We want to take advantage of SRS, and extend it to reconcile the results returned from a number of databases. We will find data repetitions and remove them first. The next step will be to provide the facility to select the information in the tree according to semantic criteria, so that the user interested in protein structures gets the structure information and a user interested in gene localisation gets a part of the XML tree showing gene locations. This grouping can be performed by placing each source database into a general category according to the information it stores. This is similar to the grouping of the databases indexed by the SRS at EBI[17]. Additionally, if a user wants to work with a list of genes, clustering operations could be carried out to highlight the genes that share subtrees of data. Finally, we need to think how to present this information in a way that supports the understanding of this complex data. This last issue remains outside the scope of this work.

6 XML Indexing and Data Integration

A high-level overview of database integration techniques is provided by Garcia-Molina and colleagues [6] who distinguish between federation, warehousing and mediation. In the biological arena, NCBI databases are a *federation* where the sources are independent, but call on one another to supply additional information. The second approach, *warehousing,* requires constructing local copies of relevant databases. SRS is a warehousing system that keeps data fresh by downloading new information overnight and re-indexing it. Lastly, *mediation* is the use of software components that support queries over data sources using various storage formats and schemas. Discovery Link [10] is a mediation system that distributes relational queries and brings the results together. Warehousing is assumed to provide more efficient queries, because the dependence on external servers and the volume of internet traffic is reduced. However, warehousing or mediation do not automatically integrate data. To integrate data, one needs to write complex queries, and none of the tools reviewed in Section 3 offer a query facility that could be used by a biologist who does not understand the semantics and syntax of the underlying data sources.

There are two facets to data integration. One is the integration of schemas and the other the integration of data values. So far, schema integration has received the most attention and recent reviews of schema matching have been conducted by Rahm and Bernstein [15] and Do and colleagues [3]. Assuming that one-to-one schema mappings can be generated, there is no accepted way to store them, manipulate them or to merge more than 2 schemas [12], unless they cover the same or very similar data [7]. Current work in the area of schema mapping does not concern itself with data values. However, there have been attempts to include data values in query view definition, most recently by Yan and colleagues [18].

[17] srs.ebi.ac.uk

Wrapping and mediation systems abstract from data values. This is partly predicated by the fact that they are designed to perform lazy access to data. Example systems include Bio-Kleisli, K2, Tambis, Discovery Link and OPM [10]. XML based approaches in this area focus on query languages and schema transformations [11,15,13,14,5].

Recent work in machine learning (ML) is beginning to address the use of data values in data integration. Kurgan and co-workers [9] performed ML on XML tags but did not consider all the data values stored in the tree. Chua and co-workers [2] performed relational data integration via attribute data matching, assuming the knowledge of entity matches. The strength of the method lies in the identification of a variety of statistical tests that are applied in attribute value matching. The method is of interest and we are planning to test it in XML data integration, by applying it to leaves that cannot be matched using relational methods. Doan *et al.* [4] use ML in concept matching. They match pairs of ontologies, which include data values stored in tree leaves. Their system uses a subset of data values to achieve concept matching and takes into account constraints, neighbourhoods of nodes, data paths and heuristics. Despite the progress reported in ML, we believe that current ML approaches suffer from several drawbacks. They can only match a pair of schemas at a time. They use only some of the data because the matching process entails an exhaustive comparison of all attributes or nodes, which leads to a combinatorial explosion of matching activity. We believe that ML could be applied as a last step in data matching and should be supported by indexing, in order to contain the combinatorial explosion of matching.

7 Integrating Data for One Gene or Protein

The semantics of XML data rest both in the tree structure and in the leaves. We intend to benefit from both types of information. To encode the tree structure, we index all paths, including the leafs. When the data is indexed, each leaf value can be examined in turn, to see if it is a candidate for matching (redundancy removal). For each leaf value we check how many times it occurs in the data, and count how many distinct paths lead to it. Groups of leaves that share the same paths are immediate candidates for path merging if they come from distinct databases. If a leaf is reachable via two distinct paths in one database, the semantics of those paths need to be examined either automatically (by algorithms) or with expert help. For any two matching paths, as identified by shared leaf sets, we can also check the immediate leaf neighbourhood to identify shared subtrees. We now outline some of the queries to use in data integration.

The simplest case is the same leaf being reachable by two different paths in two databases. This can be formalised as paths *DB1/p1/a1/leaf1* and *DB2/p2/a2/leaf1*, which can be merged given a significant number of leaves that share two such patterns. The case of matching leaves in the same database, which needs further disambiguation can be expressed as *DB1/p1/a1/leaf1* and *DB1/p2/a2/leaf1*. Shared subtrees can be captured as paths that share leaves and prefixes, i.e. *DB1/p1/a1/leaf1* and *DB1/p1/a2/leaf2*, matching the pair consisting of *DB2/p3/a3/leaf1* and *DB2/p3/a4/leaf2*, again based on a significant number of subtrees sharing the pattern. A further step would be to explore paths of type *DB1/p1/a1/leaf1*, *DB2/p2/a2/leaf5*, *DB2/p2/a2/leaf6* where *leaf1* = *leaf5* + *leaf6* (+ expresses concatenation), in order to identify trees that split longer strings into multiple leaves. Finally, linguistic data mining could be performed on leaves to identify matches that are not exact. We are planning to use techniques similar to those described by [12,4,2].

The analysis of leaf values assumes that such values are indexed with the paths leading to them. We propose to use a relational representation that will enable the identification of shared leaf sets.

8 The Indexing Algorithm

Database names are stored in a relation *DB(i, DBname)*. Each record in a database has a *Key* stored in a relation *Key(j, Keyname)*. Tags are recorded in a relation *Tag(k, Tagname)*. XML paths and their components, excluding the leaves, are indexed in relation *Path(l, k, depth)* where l is the path ID, k is the tag ID and *depth* is the distance of the tag in the path from the root. Text contained in the leaves is stored in a relation *Leaf(m, Leaftext)* and the relationship between the database i, key j, path l and leaf m, with leaf order *order* within a given record is expressed as *Data(i, j, l, m, order)*. The proposed index can be built in one text traversal, using relational technology and memory-resident tries. As the XML data is traversed, it is written line by line to the database, with appropriate tables being updated to reflect the incoming data. We will use memory-resident tries to store tags, keys, paths and leaves, with the relevant identifiers of each entry kept in the tries to avoid database lookups during the indexing phase.

The complexity of building the index can be expressed in relationship to the length of the text n. It is $O(n \log n)$ because the average tree height is $\log n$. The creation of new integer identities for *tokens* like paths, keys, leaves, etc. is done in logarithmic time because it requires one lookup of the last integer used in each domain (constant time) and one traversal from the trie root to a leaf.

9 Application Architecture

The process of data integration required by functional genomics consists of four phases. In Phase 1 an initial list of gene names is produced. This list can be downloaded from the Affymetrix database, which maps probe names to gene names. Alternatively, a search program like Mascot[18], which accepts a list of peptide masses as input and produces a list of matching proteins, can provide the protein names. In Phase 2 the query is expanded to add a list of synonyms to each gene or protein name. We will adopt the solution used by the GeneCards which

[18] www.matrixscience.com/

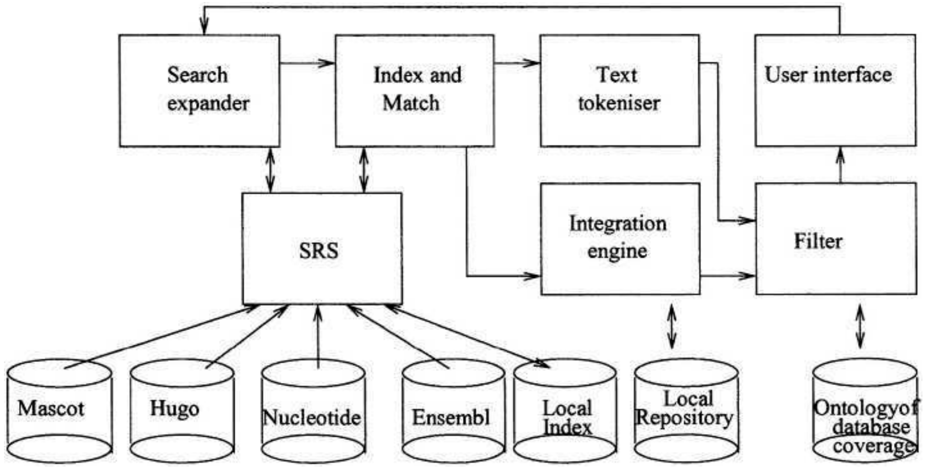


Fig. 6. Architecture for integrated search over biological databases

uses the HUGO database of gene names to expand searches with gene synonyms, but we will use synonym tables for the human, mouse and rat genomes. In Phase 3 a query is issued to the SRS system and links to individual database entries from several databases are returned. Results are returned from these databases either as objects or XML trees. All structures are converted to the common XML format prior to reconciliation, and added to the index. A local repository is used to store XML structures. In Phase 4 the data are integrated and delivered to the user. It is possible to carry out all four phases in the background and prepare a daily up-to-date digest of information for all genes of interest. When a user poses a query, the pre-processed results for each gene are brought in and further operations involving the selection of a subset of data relevant to the particular user can be performed. Figure 6 captures our system architecture for data integration.

10 Discussion, Conclusions, and Further Work

We believe we have captured a new type of user requirement in the area of biological database integration. This requirement consists of the need to acquire all relevant data about a list of gene or protein names produced by a large scale experiment. The further part of this requirement is to remove redundancies in the data and to allow for selection and clustering of the results. This requirement is hard to satisfy using either schema matching or manual data search approaches. We sketch out how this requirement could be met with a new approach to XML data integration.

To our knowledge none of the existing approaches to XML data integration has proposed to index the data leaves and examine the relationships between leaf values in the context of the paths present in the tree. Similarly, we have not seen any accounts of the use of data indexes in machine learning approaches to data integration. We propose to use a path and leaf index that can be built in $O(n \log n)$ time for a dataset of size n. The index can then be traversed with relational queries to identify leaf matches and subtree matches. Additionally, data mining can be carried out on leaf sets to identify opportunities for redundancy removal.

We propose to implement the system as an added feature on top of the SRS system which will mirror a variety of databases. The databases have to be grouped using a simple ontology of data coverage, similar to the current SRS arrangement. The SRS system will be enhanced with an indexing facility for all XML paths and a data mining facility that will reduce data duplication within XML trees for each gene of interest. The data will be filtered according to the database coverage criteria, possibly clustered with regard to genes that share publications and sequence data and presented to the user. The issues of clustering and data visualisation are outside this proposal and will achieve attention in the future.

We are implementing the system we proposed in order to test the viability of our approach. The algorithms we sketched out need significant refinement and the system architecture needs to be drawn in more detail. We will store a subset of SRS data and test our ideas in the context of mouse, rat and human genomes. We will co-index experimental data produced by the biologists and present an integrated view of the external and private data sources.

We conclude that we have identified a new data integration requirement that arises in functional genomics research. We assessed the existing tools and approaches in this area and proposed a new approach that might contribute to the discussion on XML data integration and possibly lead to a better understanding of the problems of automated data integration. We are currently implementing a system prototype and will evaluate our approach with significant amounts of data.

Acknowledgements

We thank our collaborators at the University of Glasgow: Dr T. Stein and Prof. B. Gusterson of the Division of Molecular Pathology and Cancer Sciences, Dr N. Hanlon and Prof. A. F. Dominiczak of the British Heart Foundation Glasgow Cardiovascular Research Centre, A. Jones of the Department of Computing Science. We also thank Dr C. Winchester and A. Marquet of YRING (Yoshitomi Research Institute of Neuroscience in Glasgow), at the University of Glasgow and the University of Strathclyde. This research was supported by the Medical Research Council of the UK, the Royal Society, the Carnegie Trust for the Universities of Scotland and by a Synergy grant from the University of Glasgow and the University of Strathclyde.

References

1. Achard, F. *et al.*: XML, Bioinformatics and Data Integration. Bioinformatics. **17** (2001) 115–125
2. Chua, C. E. H. *et al.*: Instance-Based Attribute Identification in Database Integration. The VLDB Journal. **12** (2003) 228–243
3. Do, H. H. *et al.*: Comparison of Schema Matching Evaluations. Proc. GI-Workshop Web and Databases, Erfurt. (2002) http://lips.informatik.uni-leipzig.de:80/pub/2002-28
4. Doan, A. *et al.:* Learning to Match Ontologies on the Semantic Web. The VLDB Journal. **12** (2003) 303–319
5. Fan, H. and Poulovassilis, A.: Using AutoMed Metadata in Data Warehousing Environments. DOLAP03, (2003) 86 – 93
6. Garcia-Molina, H. *et al.:* Database Systems - The Complete Book. Prentice Hall (2002)
7. Halevy, A. *et al.:* Crossing the Structure Chasm. CIDR-2003, (2003) www-db.cs.wisc.edu/cidr/program/p11.pdf
8. Jones, A. *et al.:* Proposal for a Standard Representation of Two-Dimensional Gel Electrophoresis Data. Comparative and Functional Genomics. **4** (2003) 492–501
9. Kurgan, L. *et al.:* Semantic Mapping of XML Tags Using Inductive Machine Learning. ICMLA02, (2002) 99–109
10. Lacroix, Z. and Crichlow, T. (ed): Bioinformatics. Managing Scientific Data. Morgan Kaufmann (2003)
11. Madhavan, J. *et al.:* Generic Schema Matching with Cupid. VLDB01, Morgan Kaufmann (2001) 49–58
12. Madhavan, J. *et al.:* Representing and Reasoning About Mappings Between Domain Models. AAAI/IAAI-02, AAAI Press, (2002) 80–86
13. McBrien, P. and Poulovassilis, A.: A Semantic Approach to Integrating XML and Structured Data Sources. CAiSE01, Lecture Notes in Computer Science, Vol 2068 (2001) 330–345
14. McBrien, P. and Poulovassilis, A.: Schema Evolution in Heterogeneous Database Architectures, A Schema Transformation Approach. CAiSE02, Lecture Notes in Computer Science, Vol 2348 (2002) 484–499
15. Rahm, E. and Bernstein, P. A.: A Survey of Approaches to Automatic Schema Matching. The VLDB Journal. **10** (2001) 334–350
16. Safran, M. *et al.:* Human Gene-Centric Databases at the Weizmann Institute of Science: GeneCards, UDB, CroW 21 and HORDE. Nucleic Acids Res. **31** (2003) 142–146
17. Tanabe, L *et al.:* MedMiner: an Internet Text-Mining Tool for Biomedical Information with Application to Gene Expression Profiling. BioTechniques. **27** (1999) 1210–1217
18. Yan, L.L. *et al.:* Data-Driven Understanding and Refinement of Schema Mappings. SIGMOD Record. **30** (2001) 485–496

Heterogeneous Data Integration with the Consensus Clustering Formalism

Vladimir Filkov[1] and Steven Skiena[2]

[1] CS Dept., UC Davis, One Shields Avenue, Davis, CA 95616
filkov@cs.ucdavis.edu
[2] CS Dept. SUNY at Stony Brook, Stony Brook, NY 11794
skiena@cs.sunysb.edu

Abstract. Meaningfully integrating massive multi-experimental genomic data sets is becoming critical for the understanding of gene function. We have recently proposed methodologies for integrating large numbers of microarray data sets based on consensus clustering. Our methods combine gene clusters into a unified representation, or a consensus, that is insensitive to mis-classifications in the individual experiments. Here we extend their utility to heterogeneous data sets and focus on their refinement and improvement. First of all we compare our best heuristic to the popular *majority rule* consensus clustering heuristic, and show that the former yields tighter consensuses. We propose a refinement to our consensus algorithm by clustering of the source-specific clusterings as a step before finding the consensus between them, thereby improving our original results and increasing their biological relevance. We demonstrate our methodology on three data sets of yeast with biologically interesting results. Finally, we show that our methodology can deal successfully with missing experimental values.

1 Introduction

High-throughput experiments in molecular biology and genetics are providing us with wealth of data about living organisms. Sequence data, and other large-scale genomic data, like gene expression, protein interaction, and phylogeny, provide a lot of useful information about the observed biological systems. Since the exact nature of the relationships between genes is not known, in addition to their individual value, combining such diverse data sets could potentially reveal different aspects of the genomic system.

Recently we have built a framework for integrating large-scale microarray data based on clustering of individual experiments [1]. Given a group of source- (or experiment-) specific clusterings we sought to identify a *consensus clustering* close to all of them. The resulting consensus was both a representative of the integrated data, and a less noisy version of the original data sets. In other words, the consensus clustering had the role of an average, or median between the given clusterings.

The formal problem was to find a clustering that had the smallest sum of distances to the given clusterings:

CONSENSUS CLUSTERING (CC): *Given k clusterings, $C_1, C_2, \ldots, C_k$, find a consensus clustering C^* that minimizes*

$$S = \sum_{i=1}^{k} d(C_i, C^*) \ . \tag{1}$$

After simplifying the clusterings to set-partitions we developed very fast heuristics for *CC* with the *symmetric difference distance* as the distance metric between partitions. The heuristics were based on local search (with single element move between clusters) and Simulated Annealing for exploring the space of solutions [1]. Practically, we could get real-time results on instances of thousands of genes and hundreds of experiments on a desktop PC.

In *CC* it was assumed that the given data in each experiment were classifiable and it came clustered. The clustering was assumed to be hard, i.e. each gene belongs to exactly one cluster (which is what the most popular microarray data clustering software yields [2]). We did not insist on the clustering or classification method and were not concerned with the raw data directly (although as parts of the software tool we did provide a number of clustering algorithms). Although clear for microarray data it is important to motivate the case that clustering other genomic data is possible and pertinent. Data classification and/or clustering is pervasive in high-throughput experiments, especially during the discovery phase (i.e. fishing expeditions). Genomic data is often used as categorical data, where if two entities are in the same category a structural or functional connection between them is implied (i.e. guilt by association). As massive data resides in software databases, it is relatively easy to submit queries and quickly obtain answers to them. Consequently, entities are classified into those that satisfy the query and those that do not; often into more than two, more meaningful categories. Even if not much is known about the observed biological system, clustering the experimental data obtained from it can be very useful in pointing out similar behavior within smaller parts of the system, which may be easier to analyze further. Besides microarray data, other examples of clustered/classified genomic data include functional classifications of proteins [3], clusters of orthologous proteins [4], and phylogenetic data clusters (http://bioinfo.mbb.yale.edu/genome/yeast/cluster).

In this paper we refine and extend our consensus clustering methodology and show that it is useful for heterogeneous data set integration. First of all, we show that our best heuristic, based on Simulated Annealing and local search, does better than a popular Quota Rule heuristic, based on hierarchical clustering. In our previous study we developed a measure, the Average sum of distances, to assess the quality of data sets integration. Some of the data sets we tried to integrate did not show any benefit from the integration. Here we refine our consensus clustering approach by initially identifying groups of similar experiments which are likelier to benefit from the integration than a general set of experiments. This is equivalent to clustering the given clusterings. After performing this step the consensuses are much more representative of the integrated data. We demonstrate this improved data integration on three heterogeneous data sets, two of microarray data, and one of phylogenetic profiles. Lastly we address the issue of missing data. Because of different naming conventions, or due to experiment errors data may be missing from the data sets. The effect of missing data is that only the data common to all sets can be used, which might be a significant reduction of the amount available. We propose a method for computational imputation of missing data, based on our consensus clustering method, which decreases the negative consequences of missing data. We show that it compares well with a popular microarray missing value imputation method, KNNimpute [5].

This paper is organized as follows. In the rest of this section we review related work on data integration and consensus clustering. We review and compare our existing methodology and the popular quota rule in Sec. 2. In Sec. 3 we present our refined method for consensus clustering, and we illustrate it on data in Sec. 4. The missing data issue is addressed in Sec. 5. We discuss our results and give a future outlook in Sec. 6.

1.1 Related Work

The topic of biological data integration is getting increasingly important and is approached by researchers from many areas in computer science. In a recent work, Hammer and Schneider [6] identify categories of important problems in genomic data integration and propose an extensive framework for processing and querying genomic information. The consensus clustering framework can be used to addresses some of the problems identified, like for example *multitude and heterogeneity of available genomic repositories (C1), incorrectness due to inconsistent and incompatible data (C8),* and *extraction of hidden and creation of new knowledge (C11).*

An early study on biological data integration was done by Marcotte et al. [7], who give a combined algorithm for protein function prediction based on microarray and phylogeny data, by classifying the genes of the two different data sets separately, and then combining the genes' pair-wise information into a single data set. Their approach does not scale immediately. Our methods extends theirs to a general combinatorial data integration framework based on pair-wise relationships between elements and any number of experiments.

In machine learning, Pavlidis et al. [8] use a Support Vector Machine algorithm to integrate similar data sets as we do here in order to predict gene functional classification. Their methods use a lot of hand tuning with any particular type of data both prior and during the integration for best results. Troyanskaya et al. [9] use a Bayesian framework to integrate different types of genomic data in yeast. Their probabilistic approach is a parallel to our combinatorial approaches.

A lot of work has been done on specific versions of the consensus clustering problem, based on the choice of a distance measure between the clusterings and the optimization criterion. Strehl et al. [10] use a clustering distance function derived from information theoretic concepts of shared information. Recently, Monti et al. [11] used consensus clustering as a method for better clustering and class discovery. Among other methods they use the *quota rule* to find a consensus, an approach we describe and compare to our heuristics in Sec. 2.2. Other authors have also used the quota rule in the past [12]. Finally, Cristofor and Simovici [13] have used Genetic Algorithms as a heuristic to find median partitions. They show that their approach does better than several others among which a simple element move (or transfer) algorithm, which coincidently our algorithm has also been shown to be better than recently [1].

It would be interesting in the near future to compare the machine learning methods with our combinatorial approach.

2 Set Partitions and Median Partition Heuristics

In this paper we focus on the problem of consensus clustering in the simplest case when clusterings are considered to be set-partitions. A *set-partition,* π, of a set $\{1,2,\dots,n\}$, is a collection of disjunct, non-empty subsets (blocks) that cover the set completely. We denote the number of blocks in π by $|\pi|$, and label them $B_1, B_2, ..., B_{|\pi|}$. If a pair of different elements belong to the same block of π then they are co-clustered, otherwise they are not.

Our consensus clustering problem is based on similarities (or distances) between set-partitions. There exist many different distance measures to compare two set-partitions (see for example [1, 14, 15]). Some are based on pair counting, others on shared information content.

For our purposes we use a distance measure based on pair counting, known as the *symmetric difference distance.* This measure is defined on the number of co-clustered and not co-clustered pairs in the partitions. Given two set-partitions π_1 and π_2, let, a equal the number of pairs of elements co-clustered in both partitions, b equal the number of pairs of elements co-clustered in π_1, but not in π_2, c equal the number of pairs co-clustered in π_2, but not in π_1, and d the number of pairs not co-clustered in both partitions. (in other words a and d count the number of agreements in both partitions, while b and c count the disagreements). Then, the symmetric difference is defined as

$$d(\pi_1,\pi_2) = b + c = \binom{n}{2} - (a+d) \ . \tag{2}$$

This distance is a metric and is related to the *Rand Index,* $R = (a+d)/\binom{n}{2}$ and other pair-counting measures of partition similarity [16]. The measure is not corrected for chance though, i.e. the distance between two independent set partitions is non-zero on the average, and is dependent on n. A version corrected for chance exists and is related to the *Adjusted Rand Index* [17]. We note that the symmetric difference metric has a nice property that it is computable in linear time [16] which is one of the reasons why we chose it (the other is the fast update as described later). The adjusted Rand index is given by a complex formula, and although it can be also computed in linear time, we are not aware of a fast update scheme for it. We will use the Adjusted Rand in Sec. 3 where the algorithm complexity is not an issue.

The *consensus clustering problem* on set-partitions is known as the *median partition problem* [18].

MEDIAN PARTITION (MP): *Given k partitions, $\pi_1, \pi_2, \ldots, \pi_k$, find a median partition π that minimizes*

$$S = \sum_{i=1}^{k} d(\pi_i, \pi) \ . \tag{3}$$

When $d(.,.)$ is the symmetric difference distance, *MP* is known to be NP-complete in general [19, 20].

In the next sections we describe and compare two heuristics for the median partition problem. In Sec. 2.1 we present briefly a heuristic we have developed recently based on a simulated annealing optimizer for the sum of distances and fast one element move updates to explore the space of set-partitions, that was demonstrated to work well on large instances of *MP*. In Sec. 2.2 we describe a popular median partition heuristic based on a parametric clustering of the distance matrix derived from counts of pairwise co-clusteredness of elements in all partitions. Our goal is to compare these two heuristics. We discuss their implementation and performance in Sec. 2.3.

The following matrix is of importance for the exposition in the next sections, and represents a useful summary of the given k set-partitions. The *consensus* or *agreement* matrix $A_{k \times k}$, is defined as $a_{ij} = \sum_{1 \le p \le k} r_{ijp}$, where $r_{ijp} = 1$ if i and j are co-clustered in partition π_p, and 0 otherwise. The entries of A count the number of times i and j are co-clustered in all k set-partitions, and are between 0 and k. Note that A can be calculated in $O(n^2 k)$ time which is manageable even for large n and k.

2.1 Simulated Annealing with One Element Moves

One element moves between clusters can transform one set-partition into any other and thus can be used to explore the space of all set-partitions. A candidate median partition can be progressively refined using one element moves. We have shown [1] that the sum of distances can be updated fast after performing a one element move on the median partition, under the symmetric difference metric:

Lemma 1. *Moving an element x in the consensus partition π from cluster B_a to cluster B_b, decreases the sum of distances by*

$$\Delta S = \sum_{\substack{all\ i \in B_a \\ i \neq x}} (k - 2a_{xi}) - \sum_{\substack{all\ j \in B_b \\ j \neq x}} (k - 2a_{xj}) \ . \tag{4}$$

Thus, once A has been calculated, updating S after moving x into a different block, takes $O(|B_a| + |B_b|)$ steps. We used this result to design a *simulated annealing* algorithm for the median partition problem. We minimize the function S (the sum of distances), and explore the solution space with a transition based on random one element moves to a random block. The algorithm is summarized below, and more details are available in [1].

Algorithm 1 Simulated Annealing, One element Move (SAOM)
Given k set-partitions π_p, $p = 1 \ldots k$, of n elements,

```
1. Pre-process the input set-partitions to obtain $A_{k\times k}$
2. "Guess" an initial median set-partition $\pi$
3. Simulated Annealing
   – Transition: Random element $x$ and a random block
   – Target Function: S, the sum, of distances
```

2.2 Quota Rule

The counts a_{ij} are useful because they measure the strength of co-clusteredness between i and j, over all partitions (or experiments). From these counts we can define distances between every pair i and j as: $m_{ij} = 1 - a_{ij}/k$, i.e. the fraction of experiments in which the two genes are not co-clustered. The distance measure m_{ij} is easily shown to be a metric (it follows from the fact that $1 + r_{acp} \geq r_{abp} + r_{bcp}$, for any $0 \leq p \leq k$). Because of that, the matrix *M* has some very nice properties that allow for visual exploration of the commonness of the clusterings it summarizes. Using this matrix as a tool for consensus clustering directly (including the visualization) is addressed in a recent study by Monti et al. [11].

The *quota rule* or *majority rule* says that elements i and j are co-clustered in the consensus clustering if they are co-clustered in sufficient number of clusterings, i.e. $a_{ij} \geq q$, for some q, usually $q \geq k/2$. This rule can be interpreted in many different ways, between two extremes, which correspond to single-link and complete-link hierarchical clustering on the distance matrix *M*. In general however any non-parametric clustering method could be utilized to cluster the n elements.

In [12] the quota rule has been used with a version of Average-Link Hierarchical Clustering known as Unweighted Pair Group Method with Arithmetic mean (UPGMA) as a heuristic for the median partition problem. Monti et al. [11] use *M* with various hierarchical clustering algorithms for their consensus clustering methodology.

Here we use the following algorithm for the quota rule heuristic:

Algorithm 2 Quota Rule (QR)
Given k set-partitions π_p, $p = 1 \ldots k$, of n elements,

```
1. Pre-process the input set-partitions to obtain $A_{k\times k}$
2. Calculate the clustering distance matrix $m_{ij} = 1 - a_{ij}/k$
3. Cluster the $n$ elements with UPGMA to obtain the consensus clustering
```

The complexity of this algorithm is on the order of the complexity of the clustering algorithm. This of course depends on the implementation, but the worst case UPGMA complexity is known to be $O(n^2 logn)$.

2.3 Comparison of the Heuristics

The SAOM heuristic was found to be very fast on instances of thousands of genes and hundreds of set-partitions. It also clearly outperformed, in terms of the quality of the consensus, the greedy heuristic of improving the sum by moving elements between blocks until no such improvement exists [1]. In the same study SAOM performed well in retrieving a consensus from a noisy set of initial partitions.

Here we compare the performance of SAOM with that of the Quota Rule heuristic. Their performance is tested on 8 data sets, five of artificial and three of real data. The measure of performance that we use is the average sum of distances to the given set-partitions, i.e. $Avg.SOD = S_{min}/(k\binom{n}{2})$, which is a measure of quality of the consensus clustering [1].

Since both heuristics are independent on k (the number of set-partitions given initially) we generated the same number of set partitions, $k = 50$, for the random data sets. The number of elements in the set-partition is $n = 10, 50, 100, 200, 500$ respectively. The three real data sets, *ccg, yst,* and *php,* are described in Section 4.1. The results are shown in Table 1, where the values are the *Avg.SOD.*

Table 1. Tests on eight data sets show SAOM yields better consensuses

ID	n	k	$QR(0.5)$	$QR(0.75)$	$SAOM$
ccg	541	173	0.141	0.145	0.139
yst	541	73	0.122	0.113	0.107
php	541	21	0.219	0.225	0.220
R_1	10	50	0.139	0.142	0.138
R_2	50	50	0.065	0.062	0.061
R_3	100	50	0.050	0.040	0.036
R_4	200	50	0.266	0.250	0.231
R_5	500	50	0.400	0.370	0.351

The SAOM was averaged over 20 runs. The QR depends on the UPGMA threshold. We used two different quotas, $a_{ij} > 0.5$, and $a_{ij} > 0.75$ which translates into $m_{ij} < 0.5$ and $m_{ij} < 0.25$, respectively.

Both heuristics ran in under a minute for all examples. In all but one of the 16 comparisons SAOM did better than QR. Our conclusion is that SAOM does a good job in bettering the consensus, and most often better than the Quota Rule.

3 Refining the Consensus Clustering

Although our algorithm always yields a candidate consensus clustering it is not realistic to expect that it will always be representative of all set-partitions. The parallel is with averaging numbers: the average is meaningful as a representative only if the numbers are close to each other.

To get the most out of it the consensus clustering should summarize close groups of clusterings, as consensus on "tighter" clusters would be much more representative than on "looser" ones. We describe next how to break down a group of clusterings into smaller, but tighter groups. Effectively this means clustering the given clusterings.

As in any clustering we begin by calculating the distance between every pair of set partitions. We use the Adjusted Rand Index (ARI) as our measure of choice, which is Rand corrected for chance [17]:

$$R_a = \frac{a+d-n_c}{\binom{n}{2}-n_c}\text{, where } n_c = \frac{(a+b)(a+c)+(c+d)(b+d)}{\binom{n}{2}} . \qquad (5)$$

Here a, b, c, and d are the pairs counts as defined in Sec. 2. The corrective factor n_c takes into account the chance similarities between two random, independent set-partitions.

Since ARI is a similarity measure we use $1 - R_a$ as the distance function. (Actually, as defined R_a is bounded above by 1 but is not bounded below by 0, and can have small negative values. In reality though it is easy to correct the distance matrix for this). The ARI has been shown to be the measure of choice among the pair counting partition agreement measures [21].

The clustering algorithm we use is average-link hierarchical clustering. We had to choose a non-parametric clustering method because we do not have a representative, or a median, between the clusters (the consensus is a possible median, but it is still expensive to compute). The distance threshold we chose was 0.5, which for R_a represents a very good match between the set-partitions [17].

Algorithm 3 Refined consensus clustering(RCC)

```
1. Calculate the distance matrix $d(\pi_1, \pi_2) = 1 - R_a(\pi_1, \pi_2)$
2. Cluster (UPGMA) the set-partitions into clusters $K_1, K_2, \dots, K_m$ with a
   threshold of $\tau = 0.5$.
3. Find the consensus partition for each cluster of set-partitions, i.e. $Cons_l =$
   $SAOM(K_l), 1 \le l \le m$
```

Note that this is the reverse of what's usually done in clustering: first one finds medians then cluster around them.

In the next section we illustrate this method.

4 Integrating Heterogeneous Data Sets

We are interested in genomic data of yeast since its genome has been sequenced fully some time ago, and there is a wealth of knowledge about it.

The starting point is to obtain non-trivial clusterings for these data sets. In the following we do not claim that our initial clusterings are very meaningful. However, we do expect, as we have shown before that the consensus clustering will be successful in sorting through the noise and mis-clusterings even when the fraction of mis-clustered elements in individual experiments is as high as 20% [1].

4.1 Available Data and Initial Clustering

For this study we will use three different data sets, two of microarray data and one of phylogeny data.

There are many publicly available studies of the expression of all the genes in the genome of yeast, most notably the data sets by Cho et al. [22] and Spellman et al. [23], known jointly as the *cell cycling genes* (**ccg**) data. The data set is a matrix of 6177 rows (genes) by 73 columns (conditions or experiments). Another multi-condition experiment, the *yeast stress* (**yst**), is the one reported in Gasch et al. [24], where the responses of 6152 yeast genes were observed to 173 different stress conditions, like sharp temperature changes, exposure to chemicals, etc.

There are many more available microarray data that we could have used. A comprehensive resource for yeast microarray data is the Saccharomyces Genome Database at Stanford (http://genome-www.stanford.edu/Saccharomyces/). Microarray data of other organisms is available from the Stanford Microarray Database, http://genome-www5.stanford.edu/MicroArray/SMD/, with ≈ 6000 array experiments publicly available (as of Jan 27, 2004).

Another type of data that we will use in our studies are phylogenetic profiles of yeast ORFs. One such data set, (**php**), comes from the Gerstein lab at Yale (available from: *http://bioinfo.mbb.yale.edu/genome/yeast/cluster/profile/sc19-rank.txt*). For each yeast ORF this data set contains the number of significant hits to similar sequence regions in 21 other organisms' genomes. A significant hit is a statistically significant alignment (similarity as judged by PSI-BLAST scores) of the sequences of the yeast ORF and another organism's genome. Thus to each ORF (out of 6061) there is associated a vector of size 21.

We clustered each of the 267 total experiments into 10 clusters by using a one dimensional version of K-means.

4.2 Results

Here we report the results of two different studies. In the first one we integrated the two microarray data sets, **ccg** and **yst**. Some of the clusters are shown in Table 4.2. It is evident that similar experimental conditions are clustered together.

The consensus clusterings are also meaningful. A quick look in the consensus of cluster 4 above shows that two genes involved in transcription are clustered together, YPR178w, a pre-mRNA splicing factor, and YOL039w, a structural ribosomal protein.

In our previous study [1] we mentioned a negative result upon integrating the 73 **ccg** set-partitions together with the 21 **php** set-partitions from the phylogeny data. The results showed that we would not benefit from such an integration, because the *Avg.SOD* = 0.3450, was very close to the Avg. SOD of a set of random set-partitions of 500 elements. In our second study we show that we can integrate such data sets more meaningfully by clustering the set partitions first. After clustering the set-partitions, with varying numbers of clusters from 3 to 10 the set-partitions from the two data sets **php** and **ccg** were never clustered

Table 2. Example Consensus Clusters of **ccg** and **yst**

Cluster 4 (Avg. SOD = 0.1179)	Cluster 5 (Avg. SOD = 0.1696)
Heat Shock 05 minutes hs-1	2.5mM DTT 015 min dtt-1
Heat Shock 10 minutes hs-1	steady-state 1M sorbitol
Heat Shock 15 minutes hs-1	aa starv 4 h
Heat Shock 20 minutes hs-1	aa starv 6 h
Heat Shock 30 minutes hs-1	YPD 4 h ypd-2
Heat Shock 40 minutes hs-1	galactose vs. reference pool car-1
Heat Shock 000 minutes hs-2	glucose vs. reference pool car-1
Heat Shock 000 minutes hs-2	mannose vs. reference pool car-1
Heat Shock 015 minutes hs-2	raffinose vs. reference pool car-1
"heat shock 17 to 37, 20 minutes"	sucrose vs. reference pool car-1
"heat shock 21 to 37, 20 minutes"	YP galactose vs reference pool car-2
"heat shock 25 to 37, 20 minutes"	Heat Shock 005 minutes hs-2
"heat shock 29 to 37, 20 minutes"	elu0

together. This is an interesting result in that it indicates that the phylogenetic profiles data gives us completely independent information from the cell cycling genes data. Similarly integrating **yst** with **php** yielded clusters which never had overlap of the two data sets. The conclusion is that the negative result from our previous study is due to the independence of the **php** vs both **ccg** and **yst**, which is useful to know for further study. Our clusters are available from *http://www.cs.ucdavis.edu/~filkov/integration.*

5 Microarray Data Imputation

Microarrays are characterized by the large amounts of data they produce. That same scale also causes some data to be corrupt or missing. For example, when trying to integrate *ccg* and *yst* we found that out of the ≈ 6000 shared genes only about 540 had no missing data in both sets. Experimenters run into such errors daily and they are faced with several choices: repeat the experiment that had the missing value(s) in it, discard that whole gene from consideration, or impute the data computationally. The idea behind data imputation is that since multiple genes exhibit similar patterns across experiments it is likely that one can recover, to a certain degree, the missing values based on the present values of genes behaving similarly.

Beyond simple imputation (like substituting 0's, or the row/column averages for the missing elements) there have been several important studies that have proposed filling in missing values based on local similarities within the expression matrix, based on different models. A notable study that was first to produce useful software, *(KNNimpute,* available at *http://smi-web.stanford.edu/projects/helix/pubs/impute/)* is [5]. More recent studies on missing data imputing are [25, 26].

We use the consensus clustering methodology to impute missing values. We demonstrate our method on microarray gene expression data by comparing it to the behavior of KNNimpute. Intuitively, the value of all the genes in a cluster in the consensus have values that are close to each other. A missing value for a gene's expression in a particular experiment is similar to having an erroneous value for the expression. In all likelihood, the gene will therefore be mis-clustered in that experiment. However, in other experiments that same gene will be clustered correctly (or at least the chance of it being mis-clustered in multiple experiments is very small). Thus identifying the similar groups of experiments and the consensus clustering between them gives us a way to estimate the missing value. We do that with the following algorithm.

Algorithm 4 Consensus Clustering Impute (CCimpute)

```
1. While clustering initially (within each experiment), for all $i$ and $j$ s.t. $v_{ij}$ is missing: place gene $i$ in a singleton cluster in the $j$-th set partition
2. Cluster (UPGMA) the set-partitions under the $R_a$ measure into clusters $K_1, K_2, \ldots, K_m$, with threshold $\tau = 0.5$
3. Find the consensus partition for each cluster of set-partitions, i.e. $Cons_l = SAOM(K_l), 1 \le l \le m$
4. Estimate a missing $v_{ij}$ as follows: Let experiment $j$ be clustered in cluster $K(j)$, and let gene $i$ be clustered in cluster $C(i)$ in $Cons_{K(j)}$. Then $\hat{v_{ij}} = \sum_{x \in C(i)} v_{xj}/|C(i)|$, i.e.
```

In other words, the missing value v_{ij} is estimated as the average value of the genes' expressions of genes co-clustered with i in the consensus clustering of the experiment cluster that contains j.

We compare the performance of our algorithm to that of KNNimpute. KNNimpute was used with 14 neighbors, which is a value at which it performs best [5]. We selected randomly 100 genes with no missing values from the **ccg** experiment above. Thus, we have a complete 100 × 73 matrix. Next, we hid at random 5%, 10%, 15%, and 20% of the values. Then, we used the normalized RMS (root mean square) error to compare the imputed to the real matrix.

The normalized RMS error, following [5] is defined as the RMS error normalized with respect to the average value of the complete data matrix. Let n_r be the number of rows in the matrix (i.e. number of genes) and n_c be the number of columns (i.e. experiments), and v_{ij} the experimental values. Then

$$NRMS = \frac{\sqrt{\frac{1}{n_r n_c}\sum_{i=1}^{n_r}\sum_{j=1}^{n_c}(v_{ij} - \hat{v_{ij}})^2}}{\frac{1}{n_r n_c}\sum_{i=1}^{n_r}\sum_{j=1}^{n_c} v_{ij}} . \tag{6}$$

The results are shown in Fig. 1. For reference a third line is shown corresponding to the popular method of imputing the row averages for the missing values. Although our algorithm is not better than KNNimpute it does do well enough for the purposes of dealing with missing values, as compared to the row-average method. Row-average on the other hand does similarly as column-average, and both do better than imputing with all zeros [5].

It is evident that the CCimpute values vary more than the results of the other methods. This may be a result of insufficient data (100 genes may be a small number). Further studies are needed to evaluate this issue.

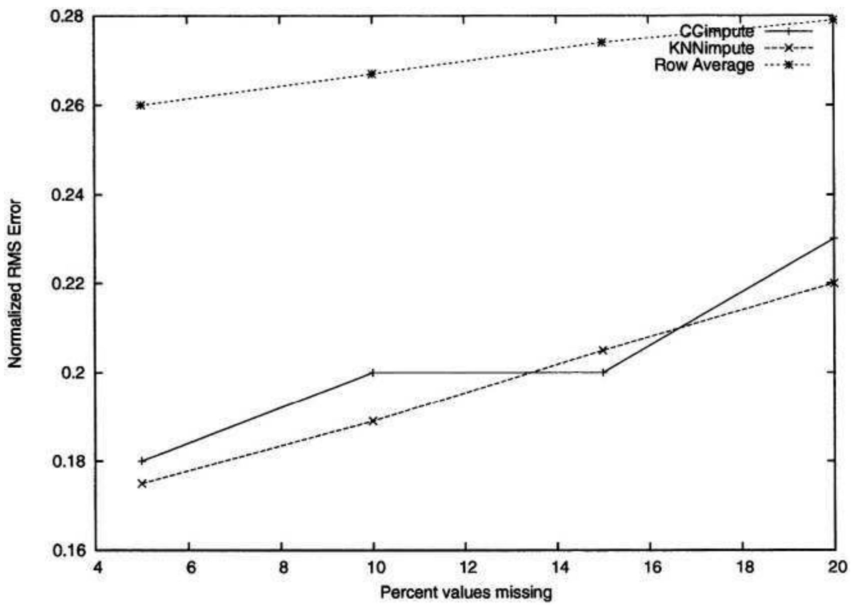


Fig. 1. Comparison of KNNimpute and CCimpute and Row Average

6 Discussion and Future

In this paper we extended our recent combinatorial approach to biological data integration through consensus clustering. We compared our best heuristic to the Quota rule with favorable results. We showed that our methods can be made more useful and biologically relevant by clustering the original clusterings into tight groups. We also demonstrated that this method can deal with missing data in the clustering.

The conclusion is that the median partition problem with the symmetric difference distance is a general method for meta-data analysis and integration, robust to noise, and missing data, and scalable with respect to the available experiments. The actual SAOM heuristic is fast (real time), and reliable, applicable to data sets of thousands of entities and thousands of experiments.

At this time we are working in two specific directions. On one hand we are trying to develop faster update methods for the Adjusted Rand measure, which is, we believe, a better way to calculate similarities between partitions. At this point our Adj. Rand methods are a thousand time slower than those on the symmetric difference metric, and hence not practical. On the other hand we are developing an integrated environment for visual analysis and integration of different large-scale data sets, which would appeal to practicing experimenters.

There is also some room for improvement of our imputation method that should make it more competitive with KNNimpute. One idea is to weigh the experimental values v_{ij}'s, which contribute to the estimated missing values, by determining how often they are co-clustered with gene i in the consensus over multiple runs of SAOM. This would effectively increase the contribution of the observed values for genes that are most often behaving as the gene whose value is missing and thus likely improve the estimate for it.

References

[1] Filkov, V., Skiena, S.: Integrating microarray data by consensus clustering. In: Proceedings of Fifteenth International Conference on Tools with Artificial Intelligence, IEEE Computer Society (2003) 418–426

[2] Eisen, M., Spellman, P., Brown, P., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Science **85** (1998) 14863–8

[3] Mewes, H., Hani, J., Pfeiffer, F., Frishman, D.: Mips: a database for genomes and protein sequences. Nucleic Acids Research **26** (1998) 33–37

[4] Tatusov, R., Natale, D., Garkavtsev, I., Tatusova, T., Shankavaram, U., Rao, B., Kiryutin, B., Galperin, M., Fedorova, N., Koonin, E.: The cog database: new developments in phylogenetic classification of proteins from complete genomes. Nucleic Acids Res **29** (2001) 22–28

[5] Troyanskaya, O., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.: Missing value estimation methods for dna microarrays. Bioinformatics **17** (2001) 520–525

[6] Hammer, J., Schneider, M.: Genomics algebra: A new, integrating data model, language, and tool for processing and querying genomic information. In: Prooceedings of the First Biennial Conference on Innovative Data Systems Research, Morgan Kaufman Publishers (2003) 176–187

[7] Marcotte, M., Pellegrine, M., Thompson, M.J., Yeates, T., Eisenberg, D.: A combined algorithm for genome wide prediction of protein function. Nature **402** (1999) 83–86

[8] Pavlidis, P., Weston, J., Cai, J., Noble, W.: Learning gene functional classifications from multiple data types. Journal of Computational Biology **9** (2002) 401–411

[9] Troyanskaya, O., Dolinski, K., Owen, A., Altman, R., Botstein, D.: A bayesian framework for combining heterogeneous data sources for gene function prediction (in s. cerevisiae). Proc. Natl. Acad. Sci. USA **100** (2003) 8348–8353

[10] Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining partitionings. In: Proceedings of AAAI, AAAI/MIT Press (2002) 93–98

[11] Monti, S., Tamayo, P., Mesirov, J., Golub, T.: Consensus clustering. Machine Learning **52** (2003) 91–118 Functional Genomics Special Issue.

[12] Gordon, A., Vichi, M.: Partitions of partitions. Journal of Classification **15** (1998) 265–285

[13] Cristofor, D., Simovici, D.: Finding median partitions using information-theoretical-based genetic algorithms. Journal of Universal Computer Science **8** (2002) 153–172

[14] Meilâ, M.: Comparing clusterings by the variation of information. In Schölkopf, B., Warmuth, M., eds.: Proceedings of the Sixteenth Annual Conference on Learning Theory(COLT). Volume 2777 of Lecture Notes in Artificial Intelligence., Springer Verlag (2003)

[15] Mirkin, B.: The problems of approximation in spaces of relations and qualitative data analysis. Information and Remote Control **35** (1974) 1424–1431

[16] Filkov, V.: Computational Inference of Gene Regulation. PhD thesis, State University of New York at Stony Brook (2002)

[17] Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification **2** (1985) 193–218

[18] Barthélemy, J.P., Leclerc, B.: The median procedure for partitions. In Cox, I., Hansen, P., Julesz, B., eds.: Partitioning Data Sets. Volume 19 of DIMACS Series in Descrete Mathematics. American Mathematical Society, Providence, RI (1995) 3–34

[19] Wakabayashi, Y.: The complexity of computing medians of relations. Resenhas IME-USP **3** (1998) 323–349

[20] Krivanek, M., Moravek, J.: Hard problems in hierarchical-tree clustering. Acta Informatica **23** (1986) 311–323

[21] Downton, M., Brennan, T.: Comparing classifications: An evaluation of several coefficients of partition agreement (1980) Paper presented at the meeting of the Classification Society, Boulder, CO.

[22] Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., Davis, R.: A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular Cell **2** (1998) 65–73

[23] Spellman, P., Sherlock, G., Zhang, M., Iyer, V., Anders, K., Eisen, M., Brown, P., Botstein, D., Futcher, B.: Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell **9** (1998) 3273–3297

[24] Gasch, A., Spellman, P., Kao, C., Carmen-Harel, O., Eisen, M., Storz, G., Botstein, D., Brown, P.: Genomic expression programs in the response of yeast cells to environment changes. Molecular Biology of the Cell **11** (2000) 4241–4257

[25] Bar-Joseph, Z., Gerber, G., Gifford, D., Jaakkola, T., Simon, I.: Continuous representations of time series gene expression data. Journal of Computational Biology **10** (2003) 241–256

[26] Zhou, X., Wang, X., Dougherty, E.: Missing-value estimation using linear and non-linear regression with bayesian gene selection. Bioinformatics **19** (2003) 2302–2307

LinkSuite™: Formally Robust Ontology-Based Data and Information Integration

Werner Ceusters[a], Barry Smith[b], James Matthew Fielding[a,b]

[a] Language & Computing nv (L&C), Hazenakkerstraat 20a, B-9520 Zonnegem, Belgium
[b] Institute for Formal Ontology and Medical Information Science, University of Leipzig, Härtelstrasse 16-18, 04107 Leipzig, Germany

Abstract. The integration of information resources in the life sciences is one of the most challenging problems facing bioinformatics today. We describe how Language and Computing nv, originally a developer of ontology-based natural language understanding systems for the healthcare domain, is developing a framework for the integration of structured data with unstructured information contained in natural language texts. L&C's LinkSuite™ combines the flexibility of a modular software architecture with an ontology based on rigorous philosophical and logical principles that is designed to comprehend the basic formal relationships that structure both reality and the ways humans perceive and communicate about reality.

1. Introduction

The integration of information resources in the life sciences is one of the most challenging problems facing bioinformatics today [1]. Researchers are flooded with information from a variety of sources and in a variety of formats, ranging from raw lab instrument data, gene expression profiles, raw sequence traces, chemical screening data, and proteomic data, to metabolic pathway models and full-fledged life science ontologies developed according to a myriad of incompatible and typically only loosely formalized schemas. Ultimately, if the robust integration of the information deriving from all of these sources is to be possible at all, tools for the formal analysis of these different types of data within a single consistent framework will have to be supplied. As a step in this direction, we describe here a methodology for information integration that is able to manipulate information scattered over many data stores while supporting a single view across the whole. The methodology can handle data stores that are owned by different organizations and located physically in different places. It can support the integration of data that are inherently heterogeneous in nature, including structured data stored in relational databases as well as data that is semi-structured via XML or HTML hyperlinking. Most importantly, it can comprehend data that is totally unstructured, including collections of text documents such as clinical discharge summaries as well as scientific journal articles.

In realizing this methodology Language and Computing nv (L&C) is working towards a framework for data-, information- and ontology-integration across all levels of generalisation and including equally information in both structured and unstructured forms. We believe that, given the complexity of the problem, the ultimate solution will be arrived at only if at least the following three tasks are dealt with in an appropriate way:

1. identifying the basic ontological foundations of a framework expressive enough to describe life science data at all levels;
2. carrying out the research in information engineering needed to create technology able to exploit this ontological framework in a way that can support the integration of massively heterogenous structured and semi-structured life science databases;
3. developing the tools for natural language understanding in the domain of the life sciences needed to extract structured data from free text documents.

L&C's LinkSuite™ environment reflects an on-going effort to implement the philosophically sound top-level ontology developed by the Institute for Formal Ontology and Medical Information Science in Leipzig [2, 3, 4]. LinkSuite™ consists of a number of mutually complementary modules, each addressing one or other of the mentioned tasks sufficiently successfully to have been granted from the technology watchers Frost and Sullivan the Healthcare Information Technology and Life Sciences Product of the Year Award during the Global Excellence in Healthcare & Life Sciences Awards Banquet in San Diego in November 2003 [5].

We first elaborate on the three tasks mentioned above. We then provide a description of the LinkSuite™ system, and finally we motivate our design choices and future directions of our research.

2. Key Requirements for Ontology-Based Information Integration

Information integration can be realised only adequately if a number of requirementsare satisfied. These fall into three categories: requirements for the sort of ontology used, requirements for the integration of structured data, and requirements for making information in text documents machine readable. We'll discuss each of these in detail.

2.1. Basic Ontological Foundations for Life Science Information Integration

Ontology is currently perceived in many circles as providing the necessary starting point for a solution to the problem of information integration both from a domain-independent perspective [6] as also in the specific field of bio-informatics [7]. We believe that ontologies can support the sort of reasoning power that is required both to optimize data-extraction from text corpora and also to optimize data-integration from a variety of disparate sources only if they rest on powerful formal-logical tools [8]. In the longer term such ontologies can also enable reasoning with the data that results from integration in ways that can open the way for large-scale hypothesis checking. But one problem continues to stand in the way of achieving these ends: terminology-oriented life science databases marked by fundamental logical inadequacies continue to evolve and expand even while incorporating ambiguities and inconsistencies with respect to such basic ontological relationships as *is-a* and *part-of* [9]. These ambiguities and inconsistencies, which result from the lack of a standard unified framework for understanding the basic relationships that structure our reality, are an obstacle to database integration and thus to the sort of computer processing of biomedical data which is the presupposition of advance in the bio-informatics field.

To rectify these problems, both formal-ontological and meta-ontological theories are required. Formal ontology is needed to provide life science applications with a set of standardized formal definitions of basic categories and relations, including the resources to deal with dependent and independent entities and with occurrents and continuants. It must have the resources to deal adequately also with the oppositions between functions and realizations (both normal and mutant genes may share the same function, but only the former can participate in those processes which are the realization of this function), and between universals and particulars (*malaria* as disease class described in textbooks versus *malaria* as particular instance of this class in this particular patient). It must also provide meta-ontological theories such as the theory of Granular Partitions [10] designed to allow navigation between ontologies in ways which can be exploited by reasoning engines. By disambiguating the terms used in the often ontologically ill-formed definitions present in most existing terminologies (cf. the misclassifications of constituents, processes and functions in the Gene Ontology [11, 12, 13] or the ontological mistakes in SNOMED-CT [14]), these formalizations may also aid in the passage of information between users and software agents. They will also help to improve consistency, both with and between ontologies, as well as contributing to the reliability of terminology curation.

2.2. Integration of (Semi-)Structured Life Science Databases

Given the basic ontological framework described above, our idea is to allow external databases (EDBs) to connect dynamically to the LinkSuite™ framework in such a way that all the implicit and explicit relationships between the data within each EDB are mapped onto relationships within the base ontology, and that the logical structure of the latter, together with the associated reasoning paper, are thereby propagated through the entire system of EDBs: the databases can be browsed and relationships between them established as if all the data were part of a single base ontology. To this end, the expressive resources of the base ontology must be sufficiently rich that we can map onto it both whole database columns and cell record data in such a way that not only the meta-data but also the incorporated instance data of the EDBs are properly apprehended. We require also that:

1. The mapping should not change the actual state of the data in either the base ontology or the EDBs, meaning that individual databases can be coupled and de-coupled at will without the mapping between the remaining EDBs and the base ontology becoming inconsistent.
2. The flow of EDB data to the ontology should occur in real-time, so that the EDBs do not need to be pre-processed in any way. The only manual intervention should be in the provision of an initial description of the structure of the database in a form that enables the latter to be mapped onto the base ontology in the appropriate way. Once this is realised, the database is dynamically mapped in such a way that all the data it contains becomes automatically accessible through the base ontology

even when updates (at least those updates that do not alter the structure of the database) subsequently appear.
3. The EDBs should continue to interoperate with external applications in the same way as they did before the mapping was effected.

2.3. Working with Textual Information Resources

At least 95% of the life science information currently publicly available resides in journals and research reports in natural language form. Even in hospitals that use an electronic medical record system, the majority of the data resides in electronic reports written in free text. Research devoted to making accessible this information has focused thus far on techniques such as document indexing and extraction of simple data elements such as dates, places, names or acronyms. Very few attempts have been made to use such techniques to obtain ontologically relevant information, for example to use automatic analysis of text documents to trigger requests for updating of ontologies such as GO or SNOMED-CT. This requires text data mining mechanisms combining statistical, linguistic and knowledge-based processing working together to overcome the problems intrinsically associated with each type of mechanism taken separately. Statistical approaches have to rely not only on a very large set of domain-related documents, but also on a reliable corpus representing non-domain-specific language usage that is needed in order to filter out what is statistically relevant in the context of the specific domain in question. Linguistic approaches can work only for processing documents in languages for which the necessary *lingware* (lexicons, grammars, machine-readable dictionary resources such as WordNet [15], and so forth) is available [16]. Such tools are certainly available for English, though not for all specialised domains, and they are available only to a limited degree for other languages. Hence a combination of different approaches is necessary, and in such a way that they complement each other mutually [17].

3. The LinkSuite™ Platform

L&C's products are based upon research initiated in the late 1980s with the objective of developing applications for natural language understanding in the healthcare domain. Ontology was identified already early on as a key presupposition of success in this regard, and the need for ontology as a language-independent representation of reality was generally well accepted in both the medical informatics [18] and natural language processing communities [19], as also was the usefulness of *situated ontologies,* i.e. ontologies that are developed for solving particular problems in specialised domains [20]. However, our experience and research convinces us that ontologies that have to operate with natural language processing applications are better suited to assist language understanding when the concepts and relations used are linguistically motivated [21]. This requirement is less important if structured data are only viewed using conventional browsers.

For these reasons, L&C has built its NLU (Natural Language Understanding) technology around a medico-linguistic ontology called LinkBase®, authored using the in-house ontology management system LinkFactory®. The applications that use these resources include TeSSI®, FreePharma® and L&C's Information Extraction Engine. In addition, L&C's OntoCreator is an NLU application that is designed to feed into LinKBase® information drawn from text documents, while MaDBoKS® feeds into LinkBase® instance data drawn from external databases. All components are developed using L&C's workflow architecture, which allows them to be plugged in or out dynamically wherever they are needed. To meet normal industrial standards of good practice for software engineering, all components are developed under a strict quality assurance process which is itself supported by its own quality assurance software and embraces product versioning, and a system for tracking and resolving errors.

LinKFactory®

LinKFactory® [22] is the proprietary L&C environment used for creating and modeling ontologies. L&C uses LinKFactory® for maintaining LinKBase®, the L&C medical ontology. This tool can use various database management systems, including Oracle and Sybase and is developed using a three-tier client-server architecture [23].

The first tier of the program runs on the user's workstation and is called the LinKFactory® Client. This tier contains a layout manager with which the user can define different frames into which he can load modules that are referred to as *beans*. Beans are Java user interfaces that facilitate communication between the user and the application and provide a visual representation of some selected portion of the underlying ontology. Examples of available beans are *concepttree* and *fulldeftree*. Beans can be assembled in a layout and linked to one another and share information by event spawning, as when the selection of a concept in the *concepttree* tells the *fulldeftree* to show the information associated with that concept in the *fulldeftree* format. The use of beans allows L&C easily to expand the functionality of LinKFactory® without disturbing the functions already supplied. Sixteen beans have been defined thus far.

The second tier of LinKFactory® runs on a server and contains the actual business logic: this tier knows what actions to perform when a user clicks a button or types in a term in a specific data capture field on his user-interface. The first and second tiers communicate through the LinKFactory® Server Interface (LSI), which can be seen as a high level API using Java RMI (remote method invocation). The LSI allows the LinKFactory® Client to pass high level requests such as 'add concept' and 'get concept' to the LinKFactory® Server's business logic layer. This tier translates a high level request (such as 'add concept') into a series of actions. In addition, it also performs authorization checks to see if the user is allowed to perform the requested actions.

The series of actions initiated by a request such as 'add concept' does not depend on any specific relational database platform because of the third tier: the data access layer, which translates Data Access Objects into relational tables containing the LinKBase® medical ontology and SQL query templates. The LinKFactory® program has been written in Java. Thus it too operates in a system-independent way and is ready to perform in a distributed network environment.

LinKBase®

LinKBase® contains over 2 million language-independent medical and general-purpose *domain-entities,* representing universals and particulars in the sense of Aristotelian ontology [24]. As such, domain-entities abstract away from the specific features of natural language representations, fulfilling to that end the same function as *concepts* in other terminologies or ontologies. They are however not equivalent to concepts, since they represent not abstractions from how humans think about real-world entities, but rather the entities themselves to which such thoughts are directed. The concepts in people's minds are clearly separated from the LinkBase® ontology proper by being represented as what are called *meta-entities,* a category which is included also in order to allow mappings to third party terminologies and ontologies. Domain-entities are associated with more than 4 million terms derived from a number of different natural language sources [25]. A *term* in this connection is a sequence of one or more *words,* which may be associated with other concepts in their turn. Domain-entities are linked together into a semantic network in which some 480 link types are used to express different sorts of relationships. The latter are derived from formal-ontological theories of mereology and topology [26, 27], time and causality [28], and also from the specific requirements of semantics-driven natural language understanding [19,29]. Link types form a multi-parented hierarchy in their own right. At the heart of this network is the formal subsumption *(is-a)* relationship, which in LinKBase® covers only some 15% of the total number of relationships involved. Currently, the system is being re-engineered in conformity with the IFOMIS theories of Granular Partitions [10] and Basic Formal Ontology [30,31].

MaDBoKS

The MaDBoKS (Mapping Databases onto Knowledge Systems) tool is an extension of Linkfactory® constructed to enable external relational databases to be connected to LinkBase® in the manner described above [32]. Database schemas from existing databases, for example from hospital patient databases or electronic patient records, can be retrieved and mapped to the ontology in such a way as to establish a two-way communication between database and ontology. The latter thereby comes to serve as a central switchboard for data integration, so that the database schemas themselves function as semantic representations of the underlying data (analogous to the semantic representations of natural language utterances that are yielded through processing by natural language understanding software). In an NLU system, a semantic parser bridges the gap between the ontology and the documents from which information is to be extracted. Here an analogous piece of software, called a *mediator,* bridges the gap between the ontology and the databases to be integrated [33]. L&C has thus far been able to successfully integrate the Gene Ontology [11], Swiss-Prot [34] and the Taxonomy database of the National Center for Biotechnology Information [35] using this approach.

TeSSI®: Terminology Supported Semantic Indexing

TeSSI® is a software application performing semantic indexing. TeSSI® first segments a document into its individual words and phrases. It then matches words and phrases in the document to corresponding LinKBase® domain-entities [36]. This step introduces ambiguity, since some entities share homonymous terms: e.g. the word *ventricle* can be used in a text to denote a *cardiac ventricle* or a *cerebral ventricle;* the phrase *short arm* may equally well denote a reduction anomaly of the upper limb or a part of a chromosome. To resolve cases of ambiguity, TeSSI® uses domain knowledge from LinkBase® to identify which domain-entity out of the set of domain-entities that are linked to a homonymous word or phrase best fits with the meaning of the surrounding words or phrases in the document.

TeSSI® then uses the matches between words and phrases identified in the document and the domain knowledge in LinkBase® to infer additional domain-entities which are only implicitly part of the subject matter of the document. The end result of this process is a graph structure whose nodes correspond to the LinkBase® domain-entities explicitly or implicitly present in the document and whose arcs correspond respectively to 1) semantic relationships derived from the LinkBase® domain ontology and 2) co-occurrence relationships derived from the position of terms in the document. The inclusion of the latter is motivated by many studies showing that co-occurring terms are likely to be also semantically related [37]. Nodes are weighted according to the number of occurrences of the corresponding terms in the document. Arcs are weighted according to the semantic distance between the corresponding entities in LinKBase® and according to the proximity of the corresponding terms in the document.

Having identified all the medical (and non-medical) terms in a document, TeSSI® then ranks the corresponding domain-entities in the order of their relevance to the document as a whole, thus identifying the topics (main subjects) of the document. Relevance scores are on a scale of 0 to 100, with 100 representing the most relevant doman-entity. To determine these scores, TeSSI® uses a constraint spreading activation algorithm on the constructed graph [38]. In this way, semantically related domain-entities reinforce each other's relevance rankings. The rationale for this algorithm stems from the observation that the domain-entities referred to by terms in any particular document will vary in their degree of semantic independence from each other. For example, a document might contain one mention each of the terms "heart failure," "aortic stenosis," and "headache", the first two being clearly more closely related to each other than to the third. An indexing system based entirely on term frequency will treat these three terms independently, thus assigning them all the same relevance. Intuitively, however, the document has twice as many mentions of heart disease as of headache. TeSSI® takes advantage of its underlying medical ontology in order to represent more accurately this type of phenomenon.

L&C's Information Extraction System

The L&C Information Extraction System consists of a number of components that successively add structured information to an unstructured text. The system takes a text document in natural language as input and creates an XML document as output. The latter then serves as the basis for further user-defined operations including querying and template-filling. As such, the information extraction system itself is an essential component in a query answering system. The XML output is created via a natural language processing procedure involving the use of Full Syntactic Parsing for syntactic analysis and LinKBase® for analysing semantics. In addition Text Grammar Analysis (TGA) is applied, which means that the system looks for relations (such as *summarization, elaboration, argumentation, exemplification,* and so forth) between parts of text. We believe that it is only through TGA carried out on top of syntactic and semantic analysis of individual sentences that a full understanding of the meaning of text in natural language will be possible in the future.

The basic components of the L&C information extraction process are:

- Segmentation
- Section Labeling
- Clause/Phrase Segmentation
- Fragment Labeling
- Information Extraction,

We deal with each of these in turn.

The input text is first segmented into paragraphs and sentences. Each sentence is then decomposed into its basic constituents, which are tagged with markers for syntactic and semantic information. Segmentation uses rules easily adaptable to the client's particular document requirements. An important step in the process is *section segmentation* carried out at whole document level rather than sentence level. A text is not an unordered succession of separate chunks of data. Rather it is a structured whole, in which each piece of information enters in at a certain functionally appropriate stage. Recognizing the different sections in a text is thus important for getting at its meaning. As an example, the first sections of this paper are: title, authors, affiliation, and abstract. In medical discharge summaries, typical initial sections are: patient-related administrative data, anamnesis, clinical findings, and so forth.

Each section is automatically assigned a label that reflects the context of the information that the section contains. Labeled sections are used to limit the scope of search when looking for particular information to be extracted. For example, discharge medication will only be looked for in sections in which discharge medication is known to appear. Labeling is based on labeling rules gathered from a training corpus that take into account a number of weighted features. Users of our system can choose whether they want to adopt an existing training corpus or create their own. If they choose the latter they are supplied with a fully customizable basic set of possible section labels with their descriptions. A graphical user interface for labeling texts is included with the system. It can be used to first label manually a training corpus that then serves as input for a supervised learning algorithm which generates in turn the rules to label similar texts automatically. Labeled sections are used to limit the scope of search when looking for particular information to be extracted. The accuracy of the labeler for medical discharge summaries amounts currently to 97.23% (tested on 4421 sections in 100 medical reports); this is an increase from the level of 96.4% achieved in 2001 [39].

Sections consist of sentences, and each sentence can be divided in its turn into clauses and phrases. To effect this division we use our Full Syntactic Parser, a hybrid system combining both symbolic and statistical approaches. We use a dependency grammar-based formalism to capture the syntactic relations between the words in a sentence, which enables us for instance to capture immediately the scope of negated elements in a sentence.

The different fragments that are recognized by the Clause/Phrase Segmenter with embedded Full Syntactic Parser receive a functional label – such as "clinical finding" or "diagnosis" – according to their content. The Fragment Labeler uses the same techniques as the Section Labeler and thus also needs to be built up by means of a training corpus, which again can be provided either by L&C or by the client. Fragment label information is used to further narrow down the amount of text in which information will be searched for.

The Information Extraction component uses information from the Section Labeler and the Fragment Labeler, as well as conceptual information from LinKBase®, syntactic information from our Full Syntactic Parser, and novel machine-learning techniques, which in combination go much further than standard text analysis algorithms relying on string matching and similar techniques.

FreePharma®

L&C developed a novel approach to formally representing and managing the information present in medication prescriptions: FreePharma® [40], The input to which is constituted by free text medication prescriptions. The latter are first parsed syntactically using full syntactic parsing aided by semantic disambiguation and statistical reinforcement: if a pure syntactic parse leads to many possible solutions, semantics and statistics are used to prune the parse tree. The syntactic parser uses semantico-syntactic labels to represent the relations between the terms in the medication prescriptions and uses various statistics to decide which analysis is the most probable. The XML-structure generated by the parser is then mapped onto a standard, pre-defined XML-template by means of semantic knowledge from a medical ontology for disambiguation and semantic slot analysis. The output of the system is thus an XML message providing a structured representation of the extracted (and initially unstructured) prescription information.

Using this formalism, we are able to gather the required information from natural language text. Because the system is not limited to structured text input, this greatly improves its flexibility and applicability. Its hybrid syntactic, semantic and statistical approach allows the system to deal with highly complex medication prescriptions.

A randomly selected corpus of 300 prescriptions, with a large coverage of possible prescription formats – including decreasing and increasing doses, as well as tapering doses and conjunctions of doses – yields a syntactic recall of 98.5%, with a syntactic precision of 96.2%. The precision of the final semantic representation amounts to 92.6%. (*Precision* here means the percentage of syntactic or semantic labels correctly assigned to terms and/or phrases in the prescriptions within the population of all the labels assigned. *Recall* refers to the percentage of correctly assigned labels with respect to the number of labels that should have been assigned.)

OntoCreator

Since so much life science information resides in journals and research reports in natural language form, it is worthwhile to develop a methodology, algorithms and software implementations that enable us to derive life-science data from free text documents and to use data extracted from these sources also in developing situated ontologies along the lines described above. OntoCreator is L&C's first and still modest attempt automatically to produce raw ontologies that can subsequently be validated and edited by users using the facilities of LinkFactory®. The module exploits the machinery described above to combine both statistical and linguistic text analysis techniques to produce raw ontologies out of text repositories covering the life sciences.

OntoCreator consists of a set of components that enable the user to analyze documents in various languages, to access and modify ontologies that are already mapped to LinKBase® and to construct graphical interfaces for accessing and editing the extracted data.

Its functionalities include the ability to:

1. extract domain-relevant terms that can be added to terminology lists, including terms not known in advance to stand in any relationship to the terms already processed;
2. propose such terms as representing domain-entities either already recognized or needing to be added to those already existing in the ontology;
3. extract semantic relationships between terms in the documents analyzed and add corresponding relations to the ontology;
4. submit the extracted terms with a relevance weight and possible semantic relationships in the form of XML documents;
5. enable the user to edit the results of the automatic ontology extraction.

4. Related Work

In [41], 53 ontology authoring systems were reviewed. Only three systems were reported to combine the functionalities of multi-user authoring, information extraction, merging of distinct ontologies and lexical support (the latter being defined by the reviewers as "*capabilities for lexical referencing of ontology elements* (*e.g., synonyms*) *and processing lexical content, e.g., searching/filtering ontology terms*)", all features that are necessary to develop and maintain the very large ontologies that will be vital to future biomedical research. Of the three, LinKFactory® is the only commercial system, the two others being OntoBuilder from the University of Savoy [42], and WebOde from the Technical University of Madrid [43]. The latter however resorts to synchronisation methods to allow multi-user access [44], and such methods are insufficient to prevent inconsistencies when two or more users are working with the same data at the same time. In addition, almost all ontology systems reviewed lack the resources to deal not only with *classes* but also with individual *instances,* i.e. entities bound to specific locations in space and time [31, 45,]. If, however, we are to incorporate instance-based data in a framework for biomedical ontology integration, then an ontology management system must go beyond what Brachman called the T-Box (of classes, or general concepts) [46] (which has served hitherto as the main focus of almost all researchers in our field) and take account also of the A-Box (containing data pertaining to the individual instances of such classes in spatio-temporal reality).

In [47] LinKBase® was reported to be the largest (in terms of number of domain-entites) medical terminology system available worldwide.

[48] describes a system that comes very close to MaDBoKS® and that is specifically designed to mediate between structured life-science databases using a much smaller ontology than LinKBase®. As is also the case for Tambis [49], however, this system is not intended to work with free-text document-based resources. Data integration in the system described in [48] is also limited to the external database schemas and does not take into account cell data.

Relevant on-going research in the combined use of structured and unstructured information sources is being conducted under the auspices of the European-funded ORIEL-project, but the currently available literature does not make it possible to assess the results obtained thus far [50].

5. Integrating Biomedical Ontologies

It is for us no surprise that Kalfoglou and Schorlemmer, after having reviewed 35 systems for their ontology mapping capacities, conclude that ontology mapping still "*faces some of the challenges we were facing ten years ago when the ontology field was at its infancy. We still do not understand completely the issues involved.*" [51]. Many researchers seem to forget that ontology is a discipline that was in its infancy not 10 but rather some 2400 years ago, when the seminal ideas of Aristotle on categories, definitions, and taxonomies were first presented – ideas which can now be seen to have enjoyed an astonishing prescience. In our view, applying philosophical and logical rigour in a way which builds on the type of realism-based analysis initiated by Aristotle is the only way to provide a coherent and unified understanding of the basic ontological distinctions required to successfully integrate the diverse domain-specific terminologies that have grown up in uncontrolled fashion in the separate parts of the biomedical informatics community [2]. Integration of heterogenous biological resources (instance-level) and integration and re-usability of bio-ontologies (class-level) are indeed the most important challenges facing the life sciences today [52]. Hence the importance of dynamic techniques such as those described above to integrate external databases with a domain-ontology.

Philosophical rigour must be applied in two equally essential dimensions. The first is in setting up the base ontology to be used as framework for integrating the separate external databases. The second is in calibrating the ontologies used in these external databases in terms of the categories and relations supplied by the base ontology.

Ontology-like structures, such as the Gene Ontology [11] and SNOMED-CT [14], are 'controlled vocabularies', i.e. they have a clean syntactic structure, which is often mistaken for a semantic structure. They consist of systems of concepts joined together via binary relations such as *is-a* and *part-of.* For the most part however, these concepts and relations are formulated only in natural language and are used in a variety of inconsistent ways even within a single ontology, and this sets obstacles in the way of ontology alignment [53]. To define a robust common structure in which ontological elements from such information resources may be mapped [2, 8, 54] thus constitutes the first dimension of philosophical rigour in the enterprise of life-science ontology integration.

The second dimension of rigour turns on the fact that mapped elements of external ontologies inherit the logical structure in which the entities and relations of the base ontology are defined and axiomatized. In this way the rigour of the base ontology is imported into external ontologies from the outside. This importation is meta-ontological, in the sense that changes designed to bring about consistency are made not directly within the external database itself, but rather via corresponding adjustments in its representation within the base ontology. This method makes it possible for us to navigate between ontologies derived from distinct external sources in consistent fashion even when the latter are not themselves consistent.

We do not thereby resolve the inconsistencies and other problems within the external ontologies themselves. While many of these problems are eliminated, or at least ameliorated, through the adoption of an approach like the one presented here, i.e. via the imposition of clear formal-ontological distinctions, it is not our intention to remodel existing databases to reflect such distinctions. Each terminology has its own purposes and advantages, and from the LinkSuite™ perspective the task of integrating the corresponding ontologies involves focusing precisely on *integration,* and not on that of *assimilation* – drawing hereby on the fact that we can attain the desired degree of consistency necessary to map these databases onto each other (by going always through the base-ontology) and adding structural information at the meta-ontological level without actual changes in the external databases themselves.

Although we have already come far, much remains to be done, especially in the area of automatic extraction of situated-ontologies from free text document collections in domains for which no formal ontology with adequate coverage thus far exists. The essential steps to be performed are: first, that of identifying terms and phrases in text documents that represent entities instantiating ontological categories such as functions and roles or dependent and non-dependent entities; and second, assessing whether or not the entities thereby found are already part of the ontology as thus far developed. Clearly these two steps must be performed in parallel, and thus some approach like that of agent-based parallel processing must be adopted, in which each agent can operate independently from the others yet is constantly generating information useful to the latter on the basis of processing that has been effected thus far while at the same time also constantly looking out for information generated by these other agents that might improve its own processing. There need to be agents that extract terms from documents that can be added to terminology lists, relate terms to already existing entities in the ontology, extract semantic relationships between entities, and request assistance from a user to assess the validity of results when no automatic mechanism can be called upon. Moreover, research must be focused around a global strategy for managing the sorts of ambiguity and uncertainty which are inevitably introduced at each decision step when an agent is deriving structured information from unstructured texts.

6. Conclusion

We have described a series of problems which arise in the study of life science ontologies, terminologies and databases, and we have sketched the design of a platform that is able to deal with them appropriately. Most problems encountered illustrate a general pattern, present in some form or another in all existing biomedical ontologies. The latter are, when assessed from the perspective of what can be achieved when an appropriate degree of formal-ontological rigour is imposed from the start, conspicuously *ad hoc* (this is the main cause of the Tower of Babel problem in current biomedical research). This *ad hoc* character is not without its history: those engaged in terminology research were forced, during the initial stages of moving from printed dictionaries and nomenclatures to digitalized information resources, to make a series of decisions about complex ontological issues – indeed about the very same issues that philosophers have pondered for millennia – at a time when the ontological nature of these issues was still not clear. To date, the importance of philosophical scrutiny in software application ontologies has been obscured by the temptation to seek immediate solutions to apparently localized problems. In this way, the forest has been lost for the trees, and the larger problems of integration have been rendered unsolvable. *Ad hoc* solutions have fostered further *ad hoc* problems.

Our research thus far, and its embodiment in LinkSuite™, constitutes a convincing demonstration of the increased adaptability that can be gained through the application of philosophical knowledge and techniques. If this success is any indicator, we have great reason to expect that further research will greatly enhance our ability to effect direct integration of further, larger and even more complex terminologies.

7. References

1 Kirsten T, Do H, Rahm E. Data Integration for Analyzing Gene Expression Data. *Proc. 2nd Biotech Day,* University of Leipzig, May 2003. 88-89.

2 Smith B., Ceusters W.: Towards Industrial-Strength Philosophy; How Analytical Ontology Can Help Medical Informatics. *Interdisciplinary Science Reviews,* 2003, 28: 2, 106-111.

3 Ceusters W, Smith B, Van Mol M. *Using ontology in query answering systems: scenarios, requirements and challenges.* In Bernardi R, Moortgat M (eds) Proceedings of the 2nd CoLogNET-ElsNET Symposium, 18 December 2003, Amsterdam, 5-15.

4 Fielding JM, Simon J, Ceusters W, Smith B. *Ontological Theory for Ontology Engineering.* In Proceedings of The Ninth International Conference on the Principles of Knowledge Representation and Reasoning, 2004 (in press)

5 Frizzell, J.: Frost & Sullivan Recognizes Language & Computing with Product of the Year Award, Nov 5, 2003. (http://awards.frost.com/prod/servlet/press-release.pag?docid=7868843&ctxixpLink=FcmCtx1&ctxixpLabel=FcmCtx2)

6 Partridge, C.: The Role of Ontology in Integrating Semantically Heterogeneous Databases. Technical Report 05/02, LADSEB-CNR Padova, Italy, June 2002

7 Lambrix, P., Habbouche, M. and Pérez, M.: Evaluation of ontology engineering tools for bioinformatics. *Bioinformatics* 19: 12, 1564-1571, 2003.

8 Ceusters W, and Smith B.: Ontology and Medical Terminology: why Descriptions Logics are not enough. *Proceedings of the Conference Towards an Electronic Patient Record* (TEPR 2003), San Antonio, 10-14 May 2003 (electronic publication).

9 Smith B. and Rosse C.: The role of foundational relations in the alignment of biomedical ontologies, *Proceedings of Medinfo,* San Francisco, 7-11 September 2004, in press.

10 Bittner, T. and Smith, B.: A Theory of Granular Partitions. In: *Foundations of Geographic Information Science,* Matthew Duckham, Michael F. Goodchild and Michael F. Worboys (eds.), London: Taylor & Francis Books, 2003, 117-151.

11 Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genet.* 25:25-29, 2000.

12 Smith B, Williams J and Schulze-Kremer S, 2003, The Ontology of the Gene Ontology, in *Biomedical and Health Informatics: From Foundations to Applications,* Proceedings of the Annual Symposium of the American Medical Informatics Association, Washington DC, November 2003, 609-613.

13 Smith B, Köhler J and Kumar A, On the Application of Formal Principles to Life Science Data: A Case Study in the Gene Ontology, in this volume.

14 Ceusters W., Smith B., Kumar A. and Dhaen C.: Mistakes in Medical Ontologies: Where Do They Come From and How Can They Be Detected? in Pisanelli DM (ed). *Ontologies in Medicine. Proceedings of the Workshop on Medical Ontologies, Rome October 2003,* Amsterdam: IOS Press, 2004 (in press).

15 Fellbaum, C. (ed.): *WordNet: An Electronic Lexical Database.* Cambridge MA: MIT Press, (1998).

16 Bod, R. and Scha, R.: Data-oriented language processing in: Young, S, and Bloothooft, G. (eds.), *Corpus-Based Methods in Language and Speech Processing,* Kluwer Academic Publishers, 137-173, 1997

17 Widdows, D.: Unsupervised methods for developing taxonomies by combining syntactic and statistical information. *Proceedings of HLT-NAACL 2003,* 197-204

18 Rector, A.L., Rogers, J.E. and Pole, P.: The GALEN High Level Ontology. In Brender J, Christensen JP, Scherrer J-R, McNair P (eds.) *MIE 96 Proceedings.* Amsterdam: IOS Press 1996,174-178.

19 Bateman, J. A.: Ontology construction and natural language. *Proc Int Workshop on Formal Ontology.* Padua, Italy, 1993: 83-93.

20 Mahesh K. and Nirenburg S.: A situated ontology for practical NLP. In *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95.* Montreal, Canada, 1995.

21 Deville, G, and Ceusters, W.: A multi-dimensional view on natural language modelling in medicine: identifying key-features for successful applications. Supplementary paper in *Proceedings of the Third International Working Conference of IMIA WG6,* Geneva, 1994.

22 Ceusters, W., Martens, P., Dhaen, C., Terzic, B.: LinKBase: an Advanced Formal Ontology Management System. Interactive Tools for Knowledge Capture Workshop, *KCAP-2001, October 2001, Victoria B.C., Canada* (http://sern.ucalgary.ca/ksi/K-CAP/K-CAP2001/).

23 Sadoski, D.: Client/Server Software Architectures--An Overview, http://www.sei.cmu.-edu/str/descriptions/clientserver_body.html, 2003.

24 Burkhardt, H. and Smith, B. (eds.): *Handbook of Metaphysics and Ontology.* Philosophia, Munich, Germany, 2 vols. (1991).

25 Montyne, F.: The importance of formal ontologies: a case study in occupational health. *OES-SEO2001* International Workshop on *Open Enterprise Solutions: Systems, Experiences, and Organizations, Rome, September 2001* (http://cersi.luiss.it/oes-seo2001/papers/28.pdf).

26 Smith, B. and Varzi A.C.: Fiat and bona fide boundaries, *Proc COSIT-97,* Berlin: Springer. 1997:103-119.

27 Smith, B.: Mereotopology: a theory of parts and boundaries, *Data and Knowledge Engineering* 1996; 20: 287-301.

28 Buekens, F., Ceusters, W. and De Moor, G.: The explanatory role of events in causal and temporal reasoning in medicine. *Met Inform Med* 1993; 32: 274-278.

29 Ceusters, W., Buekens, F., De Moor, G. and Waagmeester, A.: The distinction between linguistic and conceptual semantics in medical terminology and its implications for NLP-based knowledge acquisition. *Met Inform Med* 1998; 37(4/5): 327-33.

30 Fielding, J. M., Simon, J. and Smith. B.: Formal Ontology for Biomedical Knowledge Systems Integration. Manuscript (http://ontology.buffalo.edu/ medo/FOBKSI.pdf).

31 Grenon, P. and Smith, B.: SNAP and SPAN: Towards dynamic spatial ontology, forthcoming in *Spatial Cognition and Computation.*

32 Verschelde, J. L., Casella Dos Santos, M., Deray, T., Smith, B. and Ceusters, W.: Ontology-assisted database integration to support natural language processing and biomedical data-mining, under review.

33 Wiederhold, G. and Genesereth, M.: "The Conceptual Basis for Mediation Services (ps of Source)"; *IEEE Expert,* 12: 5, Sep.-Oct. 1997.

34 Boeckmann B., Bairoch A., Apweiler R., Blatter M., Estreicher A., Gasteiger E., Martin M. J., Michoud K., O'Donovan C., Phan I., Pilbout S., and Schneider M.: The Swiss-Prot protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res* 31: 365-370 (2003).
35 Wheeler, D. L., Chappey, C., Lash, A. E., Leipe, D. D., Madden, T. L., Schuler, G. D., Tatusova, T. A. and Rapp, B. A.: Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res* 2000 Jan 1; 28(1):10-4 (2000)
36 Jackson, B. and Ceusters, W.: A novel approach to semantic indexing combining ontology-based semantic weights and in-document concept co-occurrences. In Baud R, Ruch P. (eds) *EFMI Workshop on Natural Language Processing in Biomedical Applications, 8-9 March, 2002, Cyprus,* 75-80.
37 Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. Indexing by latent semantic analysis. *Journal of the American Society for Information Science,* 41(6), 391-407(1990).
38 Hendler, J. A.: Marker-Passing over Microfeatures: Towards a Hybrid Symbolic-Connectionist Model. *Cognitive Science* 1989 (1) 79-106.
39 Van Mol, M. and O'Donnell, M.: Automatic Recognition of Generic Structure: Medical Discharge Notices. In: *Text and Texture, Systemic Functional viewpoints on the nature and structure of text.* L'Harmattan, Paris, 2004 (in press).
40 Ceusters, W., Lorré, J., Harnie, A. and Van Den Bossche, B.: Developing natural language understanding applications for healthcare: a case study on interpreting drug therapy information from discharge summaries. *Proceedings of IMIA-WG6, Medical Concept and Language Representation,* Phoenix, 16-19/12/1999, 124-130.
41 Denny, M.: Ontology Building: A Survey of Editing Tools. http://www.xml.com/pub/a/2002/11/06/ontologies.html
42 Roche, C.: Corporate Ontologies and Concurrent Engineering. *Journal of Materials Processing Technology,* 107, 187-193, 2000.
43 Arpfrez, J. C., Corcho, O., Fernandez-Lopez, M. and Gomez-Perez, A.: WebODE: a scalable workbench for ontological engineering. *Proceedings of the First International Conference on Knowledge Capture (K-CAP) Oct. 21-23, 2001, Victoria, B.C., Canada,* 2001.
44 Anonymous. WebODE Ontology Engineering Platform. http://delicias.dia.fi.upm.es/webODE/
45 Bittner, T. and Smith, B.: Directly Depicting Granular Ontologies; presented at the 1st International Workshop on Adaptive Multimedia Retrieval, Hamburg, September 2003 (http://wings.buffalo.edu/philosophy/faculty/smith/articles/DDGO.pdf).
46 Brachman, R.: On the Epistemological Status of Semantic Networks, In Findler,N. (ed.). *Associative Networks: Representation and Use of Knowledge by Computers,* Academic Press, New York, 1979; 3-50.
47 Zanstra, P. E., van der Haring, E. J. and Cornet, R.: Introduction of a Clinical Terminology in The Netherlands, Needs, Constraints, Opportunities. National ICT Instituut in de Zorg, 2003.
48 Ben Miled, Z., Webster, Y., Li, N. and Liu, Y.: An Ontology for the Semantic Integration of Life Science Web Databases, *International Journal of Cooperative Information Systems,* Vol. 12, No. 2, June 2003.
49 Baker, P.G., Goble, C.A., Bechhofer, S., Paton, N.W., Stevens, R. and Brass, A.: An Ontology for Bioinformatics Applications, *Bioinformatics,* 15: 6, 510-520, 1999.
50 Anonymous: Online Research Information Environment for the Life Sciences. http://www.oriel.org/description.html
51 Kalfoglou, Y. and Schorlemmer, M.: Ontology mapping: the state of the art, *The Knowledge Engineering Review* 18(1), 2003.

52 Sklyar, N.: Survey of existing bio-ontologies. Technical report 5/2001, Department of Computer Science, University of Leipzig.(http://lips.informatik.uni-leipzig.de:80/pub/2001-30/en)

53 Gangemi, A., Pisanelli, D. and Steve, G.: Ontology Integration: Experiences with Medical Terminologies. In N. Guarino (ed.), *Formal Ontology in Information Systems,* 163-178. IOS Press, 1998.

54 Flett, A., Casella dos Santos, M. and Ceusters, W.: Some Ontology Engineering Processes and their Supporting Technologies, in: Gomez-Perez, A. and Benjamins, V. R. (eds.) *Ontologies and the Semantic Web, EKAW2002,* Springer 2002, 154-165.

BioDataServer: an Applied Molecular Biological Data Integration Service*

Sören Balko, Matthias Lange, Roland Schnee, and Uwe Scholz

Institute for Plant Genetics and Crop Plant Research (IPK) Gatersleben, Germany
{balko|lange|schnee|scholz}@ipk-gatersleben.de

Abstract. Nowadays, huge volumes of molecular biological data are available from different biological research projects. This data often covers overlapping and complemental domains. For instance, the *Swiss-Prot* database merely contains protein sequences along with their annotations, whereas the *KEGG* database incorporates enzymes, metabolic pathways and genome data. Due to the fact that this data complements and completes each other, it is desirable to gain a global view on the integrated databases instead of browsing each single data source itself.
Unfortunately, most data sources are queried through proprietary interfaces with restricted access and typically support only a small set of simple query operations. Apart from minor exceptions, there is no common data model or presentation standard for the query results. Consequentially, the integration of manifold heterogeneous, distributed databases has become a typical, yet challenging task in bioinformatics. Within this paper, we introduce our own approach called "BioDataServer" which is a user-adaptable integration, storage, analysis and query service for molecular biological data targeted at commercial customers.

1 Introduction

Massive research activities in life sciences, i.e. biology in general and genome research, more specifically, have led to huge amounts of molecular biological data that originates from different projects. Due to the proliferation of the Internet, this data was made electronically available to a world-wide audience. Despite various integration efforts, in the majority of all cases, these data sources are either separated from one another (i.e. contain information on few specific subjects) or are loosely coupled (e.g. through hyperlinks). Some of these data sources became very popular which can be justified by (i) data quality, (ii) coverage of the research subject or simply by the pure (iii) volume of the data itself. For the time being, we incorporated five data sources into our integration scenario: *Swiss-Prot* [2], *TrEMBL*, *OMIM*[1] [6], *BRENDA* [12], and *KEGG* [9]. These protein related data sources are of particular interest for the research activities,

* This work was supported by the German Ministry of Education and Research (BMBF) under grant number 0310621.

[1] OMIM is a trademark of the Johns Hopkins University.

here at IPK Gatersleben and serve as a proof-of-concept for the *BioDataServer* integration concept.

Though most of these data sources are separated from one another, some of them provide a weak extent of integration with other data sources. For instance, an entry in the BRENDA database may incorporate homologous enzyme sequences imported from other databases like Swiss-Prot and TrEMBL. Other data sources prefer a reference-based coupling to further data sources. For example, the KEGG database hyperlinks to other data sources, like BRENDA.

Besides their non-integrated data domains, most of these data sources do not provide standard interfaces like JDBC or ODBC. In contrast, the data is queried through proprietary interfaces, mostly intended for human interaction (e.g. through web technologies like HTML forms, CGI-scripts, *Java Server Pages* and the like)[2]. Correspondingly, the possible query terms are of restricted expressive power and stay far behind a full-featured relational query language like SQL. Though, this restricted functionality may seem sufficient for some typical queries posed by a user, it inadequately supports more advanced applications with arbitrarily complex queries.

Even worse, query results to the web interface are mostly presented in a semi-structured manner. That is, different data sources do not obey common presentation guidelines but rather define structure and layout of the data output by themselves. At least, it exists an internationally acknowledged set of identifiers for some molecular biological data. For example, the EC number uniquely characterizes protein functions across different data sources and, thus, serves as a useful integration aid.

Finally, the data sources do not share a common (e.g. relational) data model. Therefore, many homogenization tasks are to be completed to obtain a properly integrated view onto the remote data sources. Our integration approach introduces the concept of so-called *attribute sets* to model the remote data schemas in a common way. Thus, attribute sets are the best compromise for a joint canonical data model and can basically be regarded as unnormalized relations. On the one hand, we are able to define common operations w.r.t. this data model. On the other hand, this plain data model poses minor requirements on the remote data sources. Some pre-integration tasks like schema homogenization, transformation to the relational model and partial normalization can automatically be performed by the integration service while other tasks still have to be conducted manually. Below, we enumerate the requirements to our integration service by the following characteristics from a database-specific point of view.

1. The motivation of our approach is driven by the objective of an applicable service to flexibly integrate various molecular biological data sources into a local customer database.
2. The integration service must establish relations between the data in the integrated relational data model. We emphasise complete information preservation rather than avoidance of redundancies.

[2] Recently, some approaches to XML data exchange have been proposed [1].

3. Our implementation must offer wide data analysis facilities and shall provide a high-level application interface which allows to formulate complex queries against the integrated data.
4. We must guarantee a fast, reliable access onto the integrated schema that is amenable to the addition of own data sources. In addition, customers must be given the opportunity to specify well-defined fractions of the remote data sources that are to be present in the integrated schema.

This paper is organized as follows. In Section 2 we survey competing integration approaches. In Section 3 we introduce our system's architecture and motivate its design concepts by the requirements for a commercially applicable implementation. Section 4 formally introduces our data integration approach and exemplarily explains its data extraction and merging strategies. In Section 5 we briefly portray two sample web applications that interact with the integrated data stock. Section 6 concludes this paper by means of an outlook on our ongoing research issues.

2 Integration Approaches

Currently, many integration approaches exist in the field of molecular biological data. [10] describes different integration approaches, besides simple hyperlink navigation. These are *federated databases* [13], *mediators* [17], *multi database queries* [10], and *data warehouses* [8]. Most of the existing systems can be classified as one of these four basic types of integration. Existing implementations can be compared by means of five basic properties which are: (1) integration approach, (2) degree of integration, (3) materialization of the integration results, (4) supported data types, and (5) expressive power of the query operators.

The *degree of integration* is described as being tight or loose. A system is tightly coupled if all schemas of the integrated data sources are transformed into one common data model and a global schema exists. Whereas, an implementation is loosely integrated, if a mapping into a common data model was conducted, but no global schema exists. The *materialization* distinguishes materialized and view-based solutions. A materialized approach physically transfers information of all participating data sources into one global database. In contrast, a view-based implementation generates logical views onto the integrated data. The analysis of *supported data types* distinguishes between atomic types like numbers for integer-valued identifiers or strings, and complex types (sets, lists, bags, etc.) for nucleotide and amino acids. The property *query operators* characterizes the expressive power of queries sent against the integrated data stock. For example, arbitrarily complex selection predicates can be considered powerful query operators. These also include non-standard comparison in pattern matching, which are heavily required in bioinformatics. Simple single-attribute exact match queries have less expressive power and do not match the requirements of many advanced applications.

Many useful integration approaches already exist. Among the most well-known approaches are SRS [3], the Entrez system [16], the TAMBIS system [15], ISYS [14], and DiscoveryLink [5]. Table 1 classifies these systems according to the described properties. These systems are based on different data integration

	approach	integration degree	materia-lization	data types	query operators
SRS	data wareh. w/o global schema	loose	completely materialized	strings, NA & AA seq.	boolean pred., reg. exp., homology search
Entrez	data wareh. w/o global schema	loose	views (NCBI data sources)	strings, NA & AA seq.	boolean predicates homology search
TAMBIS	multi DB queries	loose	views	strings	case based qrs. ontologies
ISYS	federated DBMS	tight	views	strings	read only SQL
Disc. Link	federated DBMS	tight	views	strings	read only SQL

Table 1. Molecular Biological Integration Approaches

approaches, e.g. federated database systems (ISYS and DiscoveryLink), multi database systems (TAMBIS) and data warehouses (Entrez and SRS).

Our *BioDataServer* can be regarded as a (i) mediator-based approach that (ii) tightly couples the participating data sources in a (iii) completely materialized integrated data stock. Moreover, we support (iv) atomic types (string, integers, numbers) and partly sequence data (through BLAST coupling). Users can (v) modify the integrated data and query operations on the integrated data (vi) speed-up (no network load).

3 System Architecture

In short, our integration service must provide fast and reliable access to a customer-adaptable integrated database of molecular biological data. This data must be efficiently extractable from existing distributed, heterogeneous data sources ("remote data sources"). These condensed requirements bring forth some design decisions, we shall subsequently explain in detail.

Within the scope of database integration, one of the most important design issues is to either (i) physically integrate the remote data stocks into some materialized global database or (ii) to define a purely logical global view onto the remote data sources. Operating on physically integrated data guarantees a high availability with short response times and establishes an intermediate independence from schema changes in the remote data sources until the next update cycle. The most fundamental advantage, however, lies within the applicability of complex queries on the integrated data stock. That is, users can take advantage of a full-featured query language and its complete expressive power which broadens the application domain towards data visualization programs, data-intense statistical analysis, data mining and so forth.

On the other hand, we face the situation to likely operate on old, possibly outdated information which can only be mildered by time-consuming frequent update cycles to synchronize the integrated database with the remote data sources. However, biologists often wish to access consistent data during their long-running experiments. That is, too frequent updates are often dispensable, anyway. In turn, a non-materialized global view onto the autonomous remote data sources avoids redundancies, does not require a large local data storage, and ensures up-to-date information at any time. However, this concept establishes a strong dependence on the remote data sources in many ways. To name only a few, constant availability, short response times, stable query interfaces and robust result format must be provided at any time. Moreover, a high transaction load on the integrated schema would cause high network traffic, ultimately leading to inacceptable response times.

Though the benefits of a non-materialized global schema may seem preferable in academic prototypes, the robustness criteria of a materialized integrated schema clearly outweighs its disadvantages in an industry-ready system. In particular, in case of alterations in the remote schemas, users applications can still operate on the integrated global schema while database maintainers have time to adjust the data source accession implementations (e.g. modify data source adapters) until the next update cycle. In the light of this substantiation, the disadvantages of a materialized global schema are tolerable.

Our second design decision lies within the tightness of the integration. In contrast to the remote data sources which partially provide references to other data sources and, thus, are loosely coupled on their own, our integrated schema merges the structural information of the remote data sources into a semantically integrated schema which abrogates the boundaries of the hitherto disjoint remote schemas. On the beneficial side, applications working on the global schema, are no longer bound to the structure of each remote data source but operate on a single uniform structure. Moreover, the integrated schema establishes semantic relations between the so far incoherent data and does not leave this difficult task to the application programmer.

Thirdly, customers typically wish to correlate the integrated data with their own data stock. That is, customer data must be merged into the integrated schema, as well. More specifically, most customers do not wish to upload their confidential research results onto a remote server to perform data analysis (e.g. sequence alignment). This decision is typically justified by security and nondisclosure issues. There are less security concerns when working on an own local database server.

Moreover, each customer has different requirements regarding the integrated data. An adaptable integration concept must be able to flexibly assemble a set of relevant data sources to be present in the integrated schema. We will introduce (i) industry standard interfaces and protocols to the communication paths between the integration layer and the adapters and (ii) a formalization of the basic data extraction operations for automatic access plan generation which simplifies any adapter programming task to the very data source specific adaptations.

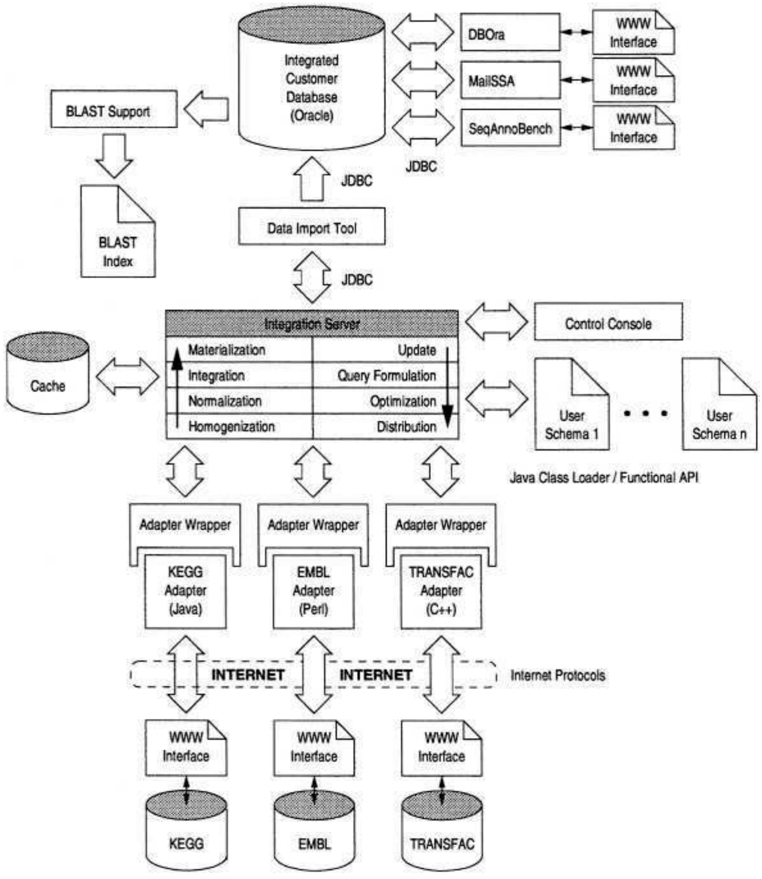


Fig. 1. System Architecture of *BioDataServer*

Our overall integration architecture is depicted in Figure 1. The remote data sources at the bottom level are distributed across the Internet. These data sources are mostly exclusively accessible via proprietary interfaces to external users. In some rare cases, remote data sources provide a relational access via industry-standard interfaces like JDBC on their own. Some other data sources can be duplicated, downloaded, and processed locally. However, an adapter usually accesses the remote data sources through a web interface and fetches its query results from a semi-structured HTML document. Whilst the query transmission can be conducted via HTTP protocols or URL assembly, gathering the result from the HTML page is merely a parsing task done by the adapter.

The integration layer runs as a server program which handles several users, each with its own integrated user schema, in parallel. That is, each user specifies its own integrated schema definition (cf. Section 4). Once, the integration layer receives an update request from a particular user, it initiates the update process which comprises (i) a formulation of data extraction requests to the remote data sources, (ii) an access plan generation and query optimization, and (iii) the distribution of the data extraction request with queries against the remote data sources.

When all data extraction requests are passed to their respective adapters, the integration layer waits for them to return their results. A cache stores intermediate results temporarily. When all adapters have finished their update tasks, the pre-integration process starts. That is, a homogenization resolves conflicting attribute names, and introduces common data types for *same*-attributes.

Altogether, the system architecture pursues a hybrid approach. On the one hand, customer applications interact with locally materialized data structured in an integrated schema. On the other hand, the integration service conceptually maintains an unrestricted number of user schemas (cf. Section 4) along with other metadata on data source capabilities and data extraction processing. This metadata is only being used to automate update requests of the integrated data as opposed to user queries that are posed against the locally materialized data stored in a relational DBMS.

4 Schema Integration

Each remote data source (RDS) may be structured in an own data model. Some data sources provide relational views onto their data while others present their query results as semi-structured text or HTML files. Therefore, prior to any data integration task those structural inconsistencies must be resolved by a transformation into a conceptual schema in our canonical data model.

In fact, the distinction between a *data source* and a *database* can be made by their architectural differences. A data source trivially is data which (1) may be distributed across several servers. Its content is (2) typically accessed through a joint web interface which only accepts proprietarily formulated queries. In contrast, a full-featured database manages data that are structured in a well-defined data model. Besides, it provides standardized query languages and interfaces like SQL and JDBC.

The main objective of the *BioDataServer* approach is to use the data in the RDS in a consistent, integrated manner. However, a loose, link-based coupling of the data sources does not serve this goal. To take full advantage of the potential of semantically related, yet physically distributed life science data, we must consider all participating RDS as one huge inter-connected information resource. State-of-the-art techniques present the RDS as inter-linked, yet isolated nodes of a global database network. Coupling rests on web technologies like HTML hyperlinks which satisfyingly support navigation but lack the ability to formulate arbitrarily complex queries like joins, grouping and aggregation, index exploitation, or similarity queries. Nevertheless, novel applications heavily rely on these advanced querying facilities when operating on life science data to perform information retrieval, knowledge discovery, or data mining tasks. In consequence, a tight database integration which incorporates the RDS into a joint data inside a state-of-the-art DBMS will be enormously profitable. Thus an integration of structurally heterogeneous, distributed, yet semantically overlapping data must be provided by our integration approach. Due to the nature of distributed information resources, our integration mechanism must cope with a high degree of autonomy w.r.t. the RDS. That is, we must typically use the pre-defined (and somewhat limited) access paths of a RDS to "extract" its content. Obviously, operations on the RDS are restricted to read-only accesses. Other aspects of data distribution and heterogeneity which have to be considered by our integration approach are (i) data fragmentation and allocation, (ii) canonical data modeling, and (iii) schema mapping and homogenization. Those aspects are being reflected by the subsequentially described schema architecture of BioDataServer (BDS).

The BioDataServer comprises three schema levels. The *Remote Export Schema* (RES) resides at the remote data sources and provides the data schemas specified in their respective native data models. In other words, the RES is a structural description of a data source. The *Adapter Schema* (AS) is a view on the RES of a data source provided in our joint canonical data model. Finally, the *Integrated User Schema* (IUS) reflects a portion of the integrated adapter schemas of the participating data sources. The IUS follows our canonical data model as well.

4.1 Remote Export Schema

Despite the fact that the majority of the remote data sources where conceptually well modeled, their majority is not directly applicable to schema integration. For instance, they typically lack an explicitely specified export schema. The main reason for this disadvantageous characteristics lies within rigorous access restrictions to the backend DBMS of a data source which prevents explicit access to its schema specification and other metadata. More precisely, schema information can be provided as a DBMS catalog, a XML schema, an IDL or other standard formats for self-describing interfaces for databases.

In some cases, there may not even be an explicitely modeled schema which would follow some formal data model specification. Instead, the data source might provide informal schema information given, for example, as textual description in natural language. In consequence, we categorize the RES provision task into two tasks: *schema re-engineering* and *schema retrieval.*

As there is no explicitely provided export schema for most data sources, a re-engineering task must be conducted to create an RES from the actual data presentation (e.g. flat files) from scratch. This time-consuming process can be accelerated through the support of semiautomatic methods. For the time being, we generate export schemas and data extraction modules by a formal grammar-based approach [4]. Alternative projects like *WebJDBS, TSIMMIS, Garlic* or *W4F* as discussed in [7], are beyond the scope of this paper. Of course, all newly constructed RES share a joint data model which corresponds to the canonical data model introduced in Section 4.2. That is, the export schema and the adapter schema match one another.

In some rare cases, the adapter can directly retrieve the catalog data of a data source through interfaces like JDBC, ODBC, SOAP/WSDL, and the like. As our implementation bases on Java, we favor the JDBC API to access data source catalogs by generated SQL statements and extract the export schema information automatically.

4.2 Adapter Schema

Adapter schemas were introduced to overcome the data model heterogeneity between export schemas. An adapter schema is specified in a joint canonical data model. The requirements for this canonical data model are twofold. On the one hand, it should support all important basic concepts that appear in a data source. On the other hand, it should be a semantically poor data model to ease the tasks of (i) schema mapping (RES to AS) and (ii) data integration (AS to IUS). Therefore, we introduce *attribute sets* as an abstraction of relations. That is, attribute sets assemble those attributes which are related, i.e. jointly presented by a data source. An attribute comprises (i) an attribute name, (ii) an attribute type, and (iii) meta information about the existence of access paths *(is indexed)* and functional dependencies *(is key)*. Thus, it widely matches the concept of relations but additionally introduces retrieval functionality and lacks foreign keys.

Hence, the syntactical notation of an adapter schema can be formally defined as follows. We call **A** an attribute out of the set of possible attributes in a database schema $\mathbf{A} = (\mathbf{N}, \mathbf{D}, \mathbf{P})$, where **N** is the attribute name, **D** its atomic data type, and **P** a combination of "is (not) key" and "is (not) indexed".

The attributes are aggregated into an *attribute set* $\mathbf{R} = \{\mathbf{A}_1, \dots, \mathbf{A}_n\}$. At the adapter level, the canonical data schema is complemented by so-called *retrieval dependencies.* In contrast to an SQL interface, retrieval from remote data sources is restricted to a set of pre-defined plain query operations. Typically, these query operations take the form (in SQL-like notation):

$$\textsf{select } A_1, \dots, A_i \textsf{ from } R \textsf{ where } B_1 = b_1 \textsf{ and} \dots \textsf{and } B_j = b_j$$

Notice, that $A_1, \dots, A_i$ and $B_1, \dots, B_j$ are (typically disjoint) sets of attributes, and $b_1, \dots, b_j$ is a list of attribute values that $B_1, \dots, B_j$ must match. The concept of *subgoals* [11] expresses this basic retrieval functionality that any adapter supports. The notation is simplified to terms like $H(B_1, \dots, B_j; A_1, \dots, A_i)$ where $B_1, \dots, B_j$ are so-called *input attributes* and $A_1, \dots, A_i$ enlist the *output attributes.* In [11] the authors propose an algorithm to efficiently answer any data extraction request by a combination of subsequent subgoals. Unfortunately, their method ends up in solving a SAT problem by some heuristic approach. Therefore, we introduce a minor adaptation which restricts the number of input attributes to 1 or 0. Or, if no input attribute is provided, the adapter provides mechanisms to retrieve a list of all output attribute values.

Retrieval dependencies with more than one input attribute appear very rarely in existing data sources. Even if one might sometimes encounter these cases, there are typically many ways to circumvent them by sequels of simple $1 : n$ retrieval dependencies. In consequence, we obtain a notable reduction of computational complexity for access plan generation (i.e. assembly of subsequently "executed" subgoals for a particular query operation), cf. Section 4.4.

In terms of a specific example, we consider the data sources KEGG and BRENDA. We abstain from an illustration of their native remote export schemas and specify their adapter schemas by means of our canonical data model. Here is a simple example: KEGG $= \{kr_1, kr_2\}$ comprises two attribute sets while BRENDA $= \{br_1\}$ contains one attribute set. The attribute sets kr_1, kr_2, and br_1 contain enzyme and pathway data:

$$\begin{aligned}
br_1 = \{&b_1 = (ec, \text{string}, \{\text{is indexed}, \text{is key}\}),\\
&b_2 = (name, \text{string}, \{\text{is not indexed}, \text{is no key}\}),\\
&b_3 = (reaction, \text{string}, \{\text{is not indexed}, \text{is no key}\}),\\
&b_4 = (organism, \text{string}, \{\text{is not indexed}, \text{is no key}\})\}\\
kr_1 = \{&a_1 = (ec, \text{string}, \{\text{is indexed}, \text{is key}\}),\\
&a_2 = (name, \text{string}, \{\text{is not indexed}, \text{is no key}\}),\\
&a_3 = (reaction, \text{string}, \{\text{is not indexed}, \text{is no key}\})\}\\
kr_2 = \{&a_4 = (map, \text{string}, \{\text{is indexed}, \text{is key}\}),\\
&a_5 = (pathwayname, \text{string}, \{\text{is not indexed}, \text{is no key}\}),\\
&a_1 = (ec, \text{string}, \{\text{is indexed}, \text{is key}\})\}
\end{aligned}$$

The attribute names *ec, name, reaction, ...,* which originate from the data sources, are replaced by the the attributes $a_1, a_2, \dots$ (KEGG) and $b_1, b_2, \dots$ (BRENDA) which appear in the AS. Notice, that some identifiers appear across many attribute sets. The retrieval dependencies in both data sources are given as follows:

$$\begin{array}{lll}
H_{\text{KEGG},1}(\ ; a_1); & H_{\text{KEGG},2}(a_1; a_2, a_3); & H_{\text{KEGG},3}(a_1; a_4);\\
H_{\text{KEGG},4}(\ ; a_4); & H_{\text{KEGG},5}(a_4; a_5, a_1); & H_{\text{BRENDA},1}(b_1; b_2, b_3, b_4);
\end{array}$$

That is, KEGG allows a retrieval of all stored EC numbers (a_1) and pathway maps (a_4). In contrast, BRENDA provides no facility to obtain a list of indexed attribute values and lacks some "entry point" for query processing.

4.3 Integrated User Schema

The IUS reflects our objective of a tight database integration where the integration component integrates the adapter schemas and maps the remote data into a materialized global database. Actually, several integrated user schemas may exist which reflect different "views" onto the integrated data. Each of these IUS covers specific data interests like enzyme and pathway information on a certain organism.

Hitherto, each adapter schema comprises its distinct set of attribute names. The IUS, however, needs to merge this information into an own set of attribute identifiers. That is, each attribute in the IUS is linked to one or more attributes in the integrated AS, i.e. their attribute values are supplied by the homogenized attribute values in the RDS. But how does the integration layer know which attributes in the participating RDS cover the same semantics? The magic is done by the concept of *same*-attributes which are manually specified by an $(n+1)$-ary relation in the IUS. More precisely, each same-relation links an attribute in the IUS with the attributes in the participating AS. In terms of our example, we model the following *same*-relations:

$$\mathsf{same}(ec, a_1, b_1); \quad \mathsf{same}(name, a_2, b_2);$$
$$\mathsf{same}(reaction, a_3, b_3); \quad \mathsf{same}(species, b_4); \quad \mathsf{same}(pathway, a_5);$$

Later on, we employ *same*-Attributes to merge attribute sets by means of join operations in the integration process. Unfortunately, molecular biological data items that express identical subjects often have different representations. This is particularly true for data that originates from different resources.

4.4 Query Processing

In the scope of this paper, the term *query processing* reflects a certain data extraction request sent against the integration layer to update the materialized data. Suppose, the user wants to update an attribute set in his integrated data. Queries against the integrated data are handled by its hosting RDBMS.

The integration layer processes a data extraction request in a series of steps, similar to ordinary database query processing. Currently, we have implemented a repository of static (i.e. manually designed) query plans to process those data extraction requests. However, progress has been made to (i) provide semi-automatism for a hitherto manual query plan design and, later on, to (ii) fully automatically process data extraction requests without any manual interaction which shall be described in this section.

First, the integration layer interpretes the query which is given in an SQL-like notation. For example, the user might request an update on an attribute set *Enzyme* which comprises the attributes *ec, name, reaction, species,* and *pathway,* for example: select $*$ from *Enzyme.*

Next, the *same*-relations come into play to identify those attribute identifiers in the AS of the RDS which appear in our query. In that case, a_1, a_2, a_3, and a_5 (KEGG) and b_1, b_2, b_3, and b_4 (BRENDA) must be retrieved from the RDS. Initially, each RDS addresses the query on its own. That is, the retrieval dependencies are being exploited to answer the query locally. The RDS establishes an empty attribute set (here $R_{\mathrm{K}}(a_1, a_2, a_3, a_5)$ and $R_{\mathrm{B}}(b_1, b_2, b_3, b_4)$) which gathers the query result. The KEGG data source facilitates the opportunity to retrieve the set of all stored values of a_1 (rule $H_{\mathrm{KEGG},1}$) and, initially, fills R_{K} as follows:

$$R_{\mathrm{K}} = \{(1.1.1.1, \_, \_, \_), (1.1.1.2, \_, \_, \_), (1.1.1.3, \_, \_, \_)\}$$

The "_" mark identifies a null value. The rule $H_{\text{KEGG},2}$ relates a_1 to a_2 and a_3. That is, for each attribute value a_1 in R_K we retrieve the appropriate a_2 and a_3 values:

$$
\begin{aligned}
R_K = \{&(1.1.1.1, \text{ADH}, \text{prim. alc.+NAD+=ald.+NADH+H+}, \_),\\
&(1.1.1.1, \text{ADH}, \text{sec. alc.+NAD+=ketone+NADH}, \_),\\
&\ldots\\
&(1.1.1.2, \text{NADP}, \text{prim. alc.+NADP+=ald.+NADPH}, \_),\\
&\ldots\\
&(1.1.1.3, \text{HSDH}, \text{L-homoser.+NAD+=L-asp. 4-semiald.+NADH}, \_),\\
&(1.1.1.3, \text{HSD}, \text{L-homoser.+NADP+=L-asp. 4-semiald.+NADH}, \_)\}
\end{aligned}
$$

Obviously, each attribute value of a_1 (EC number) yields several distinct values of a_2 (enzyme name) and a_3 (reaction). For example, the EC number "1.1.1.1" represents an enzyme abbreviated as "ADH" or "NAD" (actually, there are even more synonyms) which is involved (as substrate or product) in three chemical reactions. Thus, we obtain six combinations of enzyme names and reaction involvement for this very EC number which similarly applies to the other enzymes.

Unfortunately, there is no retrieval dependency that yields the remaining attribute a_5 (pathway name) by any of the hitherto determined attribute values (a_1, a_2, or a_3). Instead, it appears in rule $H_{\text{KEGG},5}$ which requires a_4. We follow a top-down approach to construct the retrieval dependency graph.

First, we identify all rules that yield a_5. In this simple scenario, there is only one rule where a_5 appears on the right side ($H_{\text{KEGG},5}$). In case the "input attribute" of any of these rules had been retrieved already (a_1, a_2, a_3) or there was no input attribute (like in $H_{\text{KEGG},1}$ and $H_{\text{KEGG},4}$) and no attribute was retrieved as yet, we could return this rule as one-element dependency chain and stop. In our case, however, a_4 appears as input attribute on the left side of $H_{\text{KEGG},5}$ and has not yet been retrieved. We assign a node for $H_{\text{KEGG},5}$ and recursively identify all rules where a_4 appears on the right side. In our case this is $H_{\text{KEGG},4}$ and $H_{\text{KEGG},3}$. Suppose, $H_{\text{KEGG},4}$ was identified first. In that case, a list of all attribute values of a_4 could be retrieved by the system. As we already retrieved the attribute values of a_1, a_2, and a_3, this rule is of no use since it does not properly relate a_1, a_2, a_3, and a_4. Thus, we discard $H_{\text{KEGG},4}$ and examine $H_{\text{KEGG},3}$, where a_1 appears on the left side. Obviously, a_1 is among the hitherto retrieved attributes. We, thus, assign a node for $H_{\text{KEGG},3}$, draw a directed edge between $H_{\text{KEGG},3}$ and $H_{\text{KEGG},5}$, return the emerging dependency chain $\{H_{\text{KEGG},3}, H_{\text{KEGG},5}\}$, and stop.

Although we found this simple scenario to be fairly typical for many data sources, more complex situations may arise. For example, an attribute might not be retrievable at all. In that case, the algorithm would run until it would not find a rule which has a retrievable attribute on the left side, report an error, and stop. A more difficult situation arises if the recursive traversal of dependency rules formed cycles. In our example, suppose, we arrive at $H_{\text{KEGG},3}$ where the input attribute a_1 had not yet been retrieved before. We might, thus, consult $H_{\text{KEGG},5}$ again, where a_1 appears on the right side and run into an endless loop.

To tackle this problem, we exclude any node from being visited twice. That is, any rule which has been used before (i.e. appears as a node in the dependency graph) can not be used again. The third problem occurs for dependency rules whose input attribute already appears on the right side of some other node. We may take advantage of this situation by directly establishing an edge between both nodes. However, to not run into cycles, and, thus, contradict our second postulation, we restrict this simplification to disjoint branches of the dependency graph.

Coming back to our example, we have identified a dependency chain which (i) retrieves a_4 by a_1 and then (ii) retrieves a_5 by a_4. Here a_4 serves as some temporary helper attribute (appears in brackets in R_K):

$$
\begin{aligned}
R_K = \{\ldots \\
&(1.1.1.3, \mathsf{HSDH}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, (\mathsf{MAP00260})) \\
&(1.1.1.3, \mathsf{HSD}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, (\mathsf{MAP00260})) \\
&(1.1.1.3, \mathsf{HSDH}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, (\mathsf{MAP00300})) \\
&(1.1.1.3, \mathsf{HSD}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, (\mathsf{MAP00300}))\}
\end{aligned}
$$

Due to space limitations, we have restricted ourselves to the enzyme number (a_1) "1.1.1.3" which appears in two pathways identified by the maps (a_4) "MAP00260" and "MAP00300". In the next step, this helper attribute gets replaced by a_5 through rule $H_{\mathrm{KEGG},5}$:

$$
\begin{aligned}
&R_K = \{\ldots \\
&(1.1.1.3, \mathsf{HSDH}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, \mathsf{glyc., ser., threon.\ metab.}), \\
&(1.1.1.3, \mathsf{HSD}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, \mathsf{glyc., ser., threon.\ metab.}), \\
&(1.1.1.3, \mathsf{HSDH}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, \mathsf{lys.\ biosynthesis}), \\
&(1.1.1.3, \mathsf{HSD}, \mathsf{L\text{-}homoser.{+}NAD{+}{=}L\text{-}asp.\ 4\text{-}semiald.{+}NADH}, \mathsf{lys.\ biosynthesis})\}
\end{aligned}
$$

Finally, $R_B(b_1, b_2, b_3, b_4)$ has to be filled by the attribute values of BRENDA. Unfortunately, BRENDA does not provide any mechanism to retrieve a list of all attribute values. We must, thus, provide an initial attribute list externally. Though the attribute identifiers between the KEGG and BRENDA adapter schema and the IUS are disjoint, our *same*-relations serve as a solution to this problem. In our case we should find a *same*-attribute to b_1 which appears only in the dependency rule of BRENDA, $H_{\mathrm{BRENDA},1}$. In fact, a_1 of KEGG is a *same*-attribute of b_1. We may either directly facilitate $H_{\mathrm{KEGG},1}$ to retrieve the a_1 values or, alternatively, use the distinct occurrences of a_1 from R_K, i.e.:

$$R_B = \{(1.1.1.1, \_, \_, \_), (1.1.1.2, \_, \_, \_), (1.1.1.3, \_, \_, \_)\}$$

Next, we employ $H_{\mathrm{BRENDA},1}$ to retrieve and combine the remaining attributes b_2 (name), b_3 (reaction), and b_4 (organism):

$R_\mathrm{B} = \{\ldots$
(1.1.1.3, HDH, L-homos.+NAD+=L-asp. 4-semiald.+NADH, aquif. aeolicus),
(1.1.1.3, HSD, L-homos.+NAD+=L-asp. 4-semiald.+NADH, aquif. aeolicus),
(1.1.1.3, HDH, L-homos.+NAD+=L-asp. 4-semiald.+NADH, helicob. pylori),
(1.1.1.3, HSD, L-homos.+NAD+=L-asp. 4-semiald.+NADH, helicob. pylori)}

Finally, the actual integration task must be conducted. That is, a join operation can be performed to obtain the desired enzyme information. A natural join would restrict the result to those enzymes which appear in both data sources, and, thus provide a complete information coverage for the integrated data. Alternatively, an outer join preserves all information from any data source but does eventually not provide all attribute values for each enzyme which would be expressed as null values. Based on the semantics of the application scenario, the user must choose between both alternatives. The second choice is on the join attributes. Obviously, any combination of *same*-attributes that appear in R_K and R_B may be employed. In our case a_1 (b_1), a_2 (b_2), and a_3 (b_3) fulfill this requirement. We, therefore, join on the attribute combinations (a_1, a_2, a_3) and (b_1, b_2, b_3). We (naturally) join the R_K and R_B and obtain:

Enzyme = {...
(1.1.1.3, HSD, L-homos.:NAD+=L-asp. 4-semiald.+NADH,
glyc., ser., threon. metab., aquif. aeolicus),
(1.1.1.3, HSD, L-homos.:NAD+=L-asp. 4-semiald.+NADH,
glyc., ser., threon. metab., helicob. pylori)}

For the enzyme "1.1.1.3" *(homoserine dehydrogenase)* two tuples exist in the *Enzyme* attribute set of the integrated schema. That is, R_K and R_B agree on a_1, a_2, a_3 (b_1, b_2, b_3) for these tuples, only. An alternatively applied outer join integration operation would preserve all information gathered in R_K and R_B but cause null-values in the *pathway* or *species* attributes for many tuples.

5 Sample Applications

An application demonstrating the system's capabilities should be as simple as possible in order to minimize barriers of use. This motivated the application *DBOra* that will enable the user to browse through the data. Currently, our integrated data stock comprises about 10 GByte (35 million tuples) of data stored in approximately 80 relations cross-linked through foreign keys. A second application, *MailSSA,* enables the user to select a certain starting and end table from the integrated data schema and to collect the interconnected information in between by querying the start table for a certain string or sequence and relating those data with data connected from the end table. Both applications are accessible via www-bm.ipk-gatersleben.de.

In the following we restrict ourselves to describe the functions of DBOra since it particularly highlights to afore mentioned capabilities of the integration. Deploying DBOra to navigate through the integrated data starts with either a text search over all integrated data entries or with a *BlastP* search for amino acid (AA) sequences or *BlastX* for nucleotide acid (NA) sequences, respectively. The result set will return all entries that include the initial search term(s) or provide a Blast hit of the AA or NA sequence. Hence the user can select any of those tables to display the entries more detailed. Should he find interesting data he will be able to search for data entries in other categories (tables) that might enhance and complement the already discovered data, provided that data exists of course. Furthermore, the user can restrict his search to data entries that are located in neighboring or directly connected tables with respect to the database schema of the integration system. This feature might be especially rewarding to users who formerly invested time to study the database schema and to identify relevant and irrelevant relations between the included database tables. Whereas for users investing that time, it might be more appropriate to use the option to display all information that can be reached from the current data entry by following the links between all relevant tables of the schema. The corresponding result set will be directly displayed as detailed information. With the option to display all interconnected data, all tables, neighboring and remote via corresponding neighbors, the user can search for entries in the integrated database.

6 Outlook

Finally, we outline some ideas on future progresses to improve and complement the *BioDataServer* integration system. We can basically distinguish our ongoing work into two fields which are centered around (i) the development of applications that interact with the integrated data and (ii) improvements of the adapter components.

As for the application development, we are currently investigating possibilities on how to utilize our integrated data in connection with identifying coding regions in ESTs by information retrieval mechanisms. Currently, a research project at IPK gathers information from various resources like protein domain, transcription factor, signal peptide, and subcellular localization databases to perform a statistical analysis for automatic EST annotation.

Though we focused on the overall architectural design and query processing, semi-automatic adapter implementation support plays an important role in minimizing the cost to introduce a new data source. Research in bioinformatics has come up with many existing tools to query and extract data from various popular data sources. To avoid re-implementations of these tools we currently develop a re-use concept. Though this approach permits a re-use of certain implementation components, most of the adapter programming work is still a tedious, error-prone task. We therefore, investigate learning mechanisms to detect the structure of a data source semi-automatically.

References

1. F. Achard, G. Vaysseix, and E. Barillot. XML, bioinformatics and data integration. *Bioinformatics,* 17(2):115–125, 2001.
2. B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research,* 31(1):365–370, 2003.
3. T. Etzold, A. Ulyanow, and P. Argos. SRS: Information Retrieval System for Molecular Biology Data Banks. *Methods in Enzymology,* 266:114–128, 1996.
4. A. Freier, R. Hofestädt, M. Lange, U. Scholz, and A. Stephanik. BioDataServer: A SQL-based service for the online integration of life science data. *In Silico Biology,* 2(0005), 2002. *Online Journal:* `http://www.bioinfo.de/isb/2002/02/0005/`.
5. L. M. Haas, P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope. DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal,* 40(2):489–511, 2001.
6. A. Hamosh, A. F. Scott, J. Amberger, C. Bocchini, D. Valle, and V. A. McKusick. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research,* 30(1):52–55, 2002.
7. M. Höding. *Methoden und Werkzeuge zur systematischen Integration von Dateien in Föderierte Datenbanksysteme.* Shaker Verlag, Aachen, 2000.
8. W. H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, Inc., 2 edition, 1996.
9. M. Kanehisa, S. Goto, S. Kawashima, and A. Nakaya. The KEGG database at GenomeNet. *Nucleic Acids Research,* 30(1):42–46, 2002. http://www.genome.ad.jp/kegg/.
10. P. D. Karp. A Strategy for Database Interoperation. *Journal of Computational Biology,* 2(4):573–586, 1995.
11. Steven Prestwich and Stéphane Bressan. A SAT Approach to Query Optimization in Mediator Systems. In *Proceedings of the Fifth International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2002), Cincinnati, Ohio, USA,* pages 252–259, 2002.
12. I. Schomburg, A. Chang, and D. Schomburg. BRENDA, enzyme data and metabolic information. *Nucleic Acids Research,* 30(1):47–49, 2002.
13. A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys,* 22(3):183–236, September 1990.
14. A. Siepel, A. Farmer, A. Tolopko, M. Zhuang, P. Mendes, W. Beavis, and B. Sobral. ISYS: a decentralized, component-based approach to the integration of heterogeneous bioinformatics resources. *Bioinformatics,* 17(1):83–94, 2001.
15. R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. TAMBIS: Transparent Access to Multible Bioinformatics Information Sources. *Bioinformatics,* 16(4):184–185, 2000.
16. T. A. Tatusova, I. Karsch-Mizrachi, and J. A. Ostell. Complete genomes in WWW Entrez: data representation and analysis. *Bioinformatics,* 15(7/8):536–543, 1999.
17. G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer,* 25(3):38–49, March 1992.

COLUMBA: Multidimensional Data Integration of Protein Annotations

Kristian Rother[1], Heiko Müller[2], Silke Trissl[2], Ina Koch[3], Thomas Steinke[4], Robert Preissner[1], Cornelius Frömmel[1], Ulf Leser[2]

[1] Universitätskrankenhaus Charité Berlin, Institut für Biochemie, Monbijoustr. 2
10098 Berlin, Germany
{kristian.rother, robert.preissner, cornelius.froemmel}@charite.de
[2] Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6,
10099 Berlin, Germany
(hmueller, trissl, leser}@informatik.hu-berlin.de
[3]Technische Fachhochschule Berlin, Seestr. 64, 13347 Berlin, Germany
ina.koch@tfh-berlin.de
[4]Zuse Institut Berlin, Takustrasse 7, Berlin, Germany
steinke@zib.de

Abstract. We present COLUMBA, an integrated database of protein annotations. COLUMBA is centered around proteins whose structure has been resolved and adds as much annotations as possible to those proteins, describing their properties such as function, sequence, classification, textual description, participation in pathways, etc. Annotations are extracted from seven (soon eleven) external data sources. In this paper we describe the motivation for building COLUMBA, its integrational architecture and the software tools we developed for the integrated data sources and keeping COLUMBA up-to-date. We put special focus on two aspects: First, COLUMBA does not try to remove redundancies and overlaps in data sources, but views each data source as a proper dimension describing a protein. We explain the advantages of this approach compared to a tighter semantic integration as pursued in many other projects. Second, we highlight our current investigations regarding the quality of data in COLUMBA by identification of hot spots of poor data quality.

1 Introduction

In life science research, there is an overwhelming amount of data in public and commercial databases available for data analysis and knowledge discovery. The time and cost effective usage of these data is hampered by two main problems: (i) the distribution of relevant data over many heterogeneous data sources and (ii) the quantity of errors and omissions within these sources. The first problem is solved by data integration approaches, while the second problem is tackled by means of data cleansing.

COLUMBA is a database of integrated protein annotations. Therefore, it has to cope with both types of problems. First, the sources currently integrated into COLUMBA are spread world-wide, are hosted on a variety of different platforms, and each has its own proper schema or format, semantically and syntactically distinct from all others.

Second, the sources suffer from incompleteness and sometimes store redundant results, which need to be identified and inconsistencies need to be removed. Within this paper, we explain our specific solutions to both problems.

Data integration in general is complicated by technical, syntactical, and semantic heterogeneity of the sources, our inability to define a global, all-embracing schema for the domain due to incomplete domain knowledge and the pure complexity of the problem, frequent changes to source data and source schemas, and the effort necessary for integrating additional data sources. Data integration systems aim at providing unified access to a set of heterogeneous data sources based on an integrated view of the data sources. For COLUMBA we choose a multidimensional integration approach, where the data are materialized in a local relational database and the global schema is built of linked component schemas for each of the sources, closely following their proper concepts. The schema is centered on protein structures, taken from the central repository, the Protein Data Bank (PDB) [1]. Apart from the PDB, COLUMBA currently integrates six different data sources, describing various aspects of proteins and protein structures.

We call our approach "multidimensional" because it treats each data source as an essentially independent dimension, describing proteins. Keeping the schema close to the concepts of the sources and connecting them by relations has the advantage of (i) presenting all data to the biological user in terms of well-known concepts, which would get blurred by semantic integration, (ii) relieving us of the necessity for semantic integration, (iii) enabling the re-usage of existing parser software for the sources, (iv) effortless addition and removal of sources due to modular design of the schema, (v) simplifying the reconstruction of the origin of each piece of information (data provenance) resulting in higher user confidence and an ease of the update process in case of source changes, and (vi) a simple and intuitive query model (and resulting web-interface) following a 'query refinement' paradigm resembling common manual procedures of the biological domain expert.

Many data sources in the biomedical domain are renowned for containing data of sometimes poor quality [2][3]. This is due to the experimental nature of the field, the quickly changing knowledge landscape, the high redundancies in experiments performed often leading to contradicting results, and the difficulties in properly describing the results of an experiment in a domain as complex as molecular biology. Furthermore, it was often observed that data quality problems multiply when data of low quality are integrated and re-used for annotation [4].

In COLUMBA, we pay special attention to the aspect of measuring data quality and detecting hot-spots of poor quality. We approach the problem by analyzing contradicting values in the case of duplicate protein entries. In COLUMBA, such duplicates do not appear at the data sources, which are considered independent, but in the core data, i.e. the PDB entries, itself. Currently there are three instantiations of the PDB, which are derived from each other, but with different procedures for data completion and correction applied. For COLUMBA we face the problem of having to choose the best origin for each single attribute value.

With this paper we want to share our experiences gained in building an integrated database in the life science domain and highlight the benefits and difficulties of the multidimensional integration approach. Further, we report on preliminary results in identifying and quantifying quality problems in complex domains such as life science.

The structure of the paper is as follows. In the next section we give an overview of the biological background and motivation for an integrated database of protein structure annotation. In Section 3 we briefly describe the characteristics and formats of the data sources integrated into COLUMBA. Section 4 describes the multidimensional data integration architecture. In Section 5 we describe the design of the database schema and integration pipeline used to populate and update the database. We also outline the possibilities for initial data cleansing during this process. In Section 6 we describe the web-interface for building and executing queries to the database. Section 7 discusses related work. We conclude in Section 8.

2 Biological Background

Researchers on protein structures are often interested in sets of proteins, sharing certain properties such as sub-folds, protein function, source organism, or pathways. Being able to generate such sets quickly has at least two major applications. First, researchers may select such a set to perform some special analysis only on the structures within this set, trying to find set-specific properties. Second, researchers have defined groups of proteins and try to correlate these groups according to the properties of the protein structures contained.

Sets of structures are mainly required for the prediction of the docking of small molecules, protein folding and protein-protein interactions, and analyzing functional relationships. All three require data from comparative studies, also carried out on specific datasets. Depending on the kind of study and interest of the research group a spectrum of very different questions arise. Examples are:

- Get all structures having the *TIM-barrel fold* with a resolution better than 2.0 Å.
- Get all structures of antibodies of the *IgG class.*
- Get all structures of enzymes in the *fatty acid degradation* pathway.
- Get all structures from *saccharomyces cerevisiae (yeast)* with less than 90% sequence similarity to a human protein.

These questions have in common that they result in a list of protein structures. The second kind of analysis, e.g., looking for a new structural motif, requires such a list as input, which is then further characterized. The criteria used for characterizing them are similar as in the above examples, because some biases in the source data have to be omitted.

We give one more elaborate example for the types of queries we pursue with COLUMBA. Proteins act together in pathways. For pathways, there are specialized databases such as KEGG [5]. However, the question is whether all pathways known today are equally well represented in terms of structures of the participating proteins. To get an impression of both, the numbers and the heterogeneity in the database, the coverage of metabolic pathways by structures was analyzed, using the integrated data in COLUMBA from the PDB and KEGG. A great mixture of coverage can be observed among pathways, some highly covered while others have no structure assigned to them at all (Table 1).

Table 1. Coverage of metabolic pathways with structures in PDB. Each pathway - a group of connected biochemical reactions - consists of several enzymes. For each enzyme, zero to many protein structures are available. If no structure for an enzyme exists, this is mainly due to experimental reasons. We analyzed to what extent pathways are already resolved by structures. This kind of query is available as a predefined button in the COLUMBA web interface

Pathway	Enzymes	Structures	Enzymes with 1+ structures
Carbon fixation	23	323	87%
Fatty acid biosynthesis II	7	32	57%
Val/Leu/Ile biosynthesis	15	38	53%
Citrate cycle (TCA cycle)	23	157	52%
Peptidoglycan biosynthesis	16	35	50%
Alkaloid biosynthesis I	34	96	15%
Ubiquinone biosynthesis	10	0	0%

3 Data Sources

COLUMBA currently integrates data from seven data sources: The Protein Data Bank (PDB), the Structural Classification of Proteins (SCOP) [6], the Kyoto Encyclopedia of Genes and Genomes (KEGG), the Dictionary of Protein Secondary Structure (DSSP) [7], the Enzyme Nomenclature Database (ENZYME) [8], the Dictionary of Interfaces in Proteins (DIP) [9], and the Protein Topology Graph Library (PTGL). Four more data sources are already physically integrated, but not yet released into the public version of COLUMBA, i.e., protein classification from the Hierarchic Classification of Protein Domain Structures (CATH) [10], functional annotation integrated from SWISS-PROT [11], the NCBI taxonomy database of organisms [12], and the Gene-Ontology (GO) [13].

Additional data sources, which we are planning to integrate soon, are the non-redundant lists of structures from PISCES [14], protein families from SYSTERS [15], and biochemical pathway charts from Boehringer-Mannheim [16].

The Protein structure database PDB is often referred to as a worst case scenario of biological data representation. We will illustrate five typical problems occurring with the PDB and other biological databases as well: (i) Starting in the 1980's as a repository of a few ASCII files of x-ray protein structures, the PDB has grown to the size of more than 22000 structures by late 2003. However, the file format originally inspired from punch cards has changed only once in those 20 years and is very difficult to parse. (ii) It contains structures not only of proteins, but also DNA, peptide fragments and large biological complexes. (iii) The structures have been resolved with several experimental methods, e.g., NMR ensembles and very low resolution electron microscopy images. The quality of the results therefore varies greatly. (iv) The data are highly redundant; there are more than 800 structures of the protein *Lysozyme,* but only one of *Photosystem I.* (v) The annotation is mostly incomplete, erroneous, and full of typographical errors. Controlled vocabularies are not applied in the database until today (but sometimes in submission tools). Thus, one has to pick relevant information very carefully from a data pool with a certain level of noise.

The data sources are available in various formats. The predominant format are flat files, e.g., for PDB, SCOP, CATH, and ENZYME, which are parsed to create load files for the target relational database management system. Boehringer and KEGG were parsed from web sites using our own parsers. Several other sources are available as database dump files, e.g., SWISS-PROT, GeneOntology, and NCBI data [17]. SWISS-PROT data are also provided as flat files and database load files. There exist public parsers for PDB, SCOP, ENZYME, and SWISS-PROT. The remaining data sources in flat file format follow a predominantly simple file format making implementation of the parser fairly simple. DSSP is a special case, because it is a program to compute the secondary structure for protein sequences from the PDB entries. PDB is also available in different formats and database versions. In Section 5.3 we shall give more information how PDB is parsed and transformed to form the basis of COLUMBA.

We integrate everything within a relational database. At time of writing the COLUMBA database contains annotation for 22453 protein structures, containing a total of 45037 polypeptide chains, 25920 biological compounds and 217015 small molecular groups. Of these entries, 89% have been annotated by at least one type of data source (sequence, function or fold), 36% by all three.

4 Multidimensional Data Integration

Defining a global schema covering semantically and structurally heterogeneous data sources poses a major challenge for the design of any data integration system. The approach most commonly described in literature is schema integration. Schema integration is defined as the activity of integrating the schema of existing data sources into a global schema by unifying the representation of semantically similar information that is represented in a heterogeneous way across the individual sources [18]. This involves the identification of semantically equivalent attributes or concepts within the different sources and the definition of a strategy for merging them into a resulting schema, covering all the aspects of the contributing sources. Recently, a number of tools have been proposed which can aid the user in this task by analyzing names and relationships of schema elements [19][20].

Within COLUMBA we use a different approach, which we call multidimensional data integration. The name is inspired from data warehouse design, in data warehousing, facts, e.g., sales, are described by dimensions, such as store, product, or customer [21]. The resulting schema is called star or snowflake schema in correspondence with the visual appearance. In analogy we center our schema on protein structures and describe them by dimensions such as function, description, classification, or sequence. Furthermore, we strictly follow the design paradigm that each data source is mapped into one dimension. Data from different sources are never mixed with each other.

The two main characteristics of our multidimensional data integration approach are (more details are given below):

- We leave the data in a schema closely following the concepts and design of the source schema.

- We model COLUMBA as a multidimensional schema, where the integrated single sources take the role of (often complex and hierarchically structured) dimensions to form a snowflake-like schema centered on the PDB entry.

4.1 System Architecture

The integration process as shown in Fig. 1 is divided into horizontal and vertical integration flow. In the first step for each of the sources a horizontal integration module performs the transformation of the original data into an initial relational representation (initial schema). The complexity of this step depends on the data source. The relational representation filled in this step is completely independent from the final COLUMBA schema.

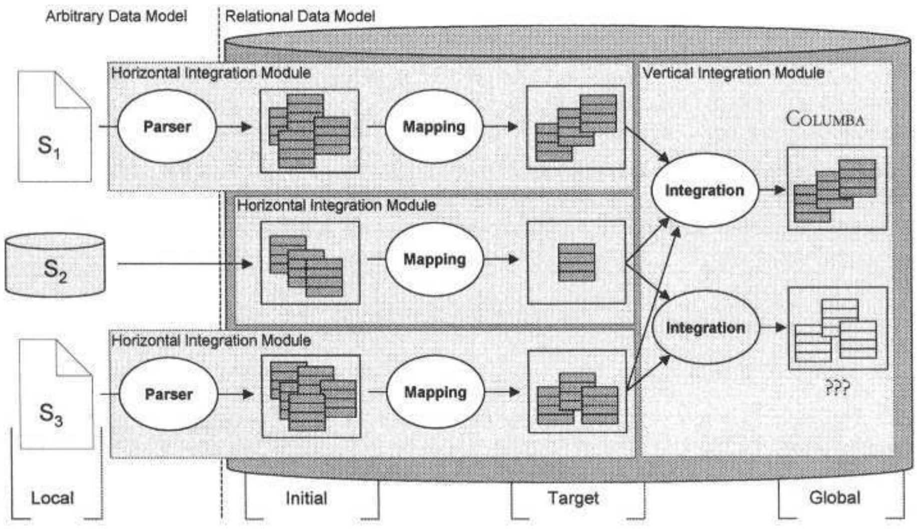


Fig. 1. Multidimensional data integration architecture showing the horizontal integration modules for each of the sources transforming the data from local schema to initial schema and target schema and the vertical integration module integrating the data within the global schema

In the second step, we define a mapping from the initial relational schema into the COLUMBA target schema. Again, this step is performed for each source, i.e., both mappings and the target schema are source-specific. The target schema can differ from the initial schema for many reasons. First, only data relevant for COLUMBA are taken into the target schema. Second, the target schema is modeled such that typical queries can be computed with sufficient performance, e.g., it is often more or less normalized than the initial schema. Third, during the mapping, sometimes also records (and not only tables) are filtered because they describe data items outside the scope of COLUMBA.

In the third step, vertical integration is performed by connecting the data in the target schemas to each other and to the PDB core by filling the linkage tables. This can be done either by using cross-link information from the sources or by performing computations built on knowledge of biological rules. In Section 5 we give a more detailed overview about the actual integration process of COLUMBA.

Having sources within the relational model enables us to define other global schemas as composition of the different sources, depending on the application requirements. We view COLUMBA as only one of the possible target applications.

4.2 Advantages of Multidimensional Data Integration

Multidimensional data integration has advantages regarding system maintenance and development, data provenance, keeping the data up-to-date, and usability. In the following sub-sections we give a detailed overview of the advantages within each of the mentioned areas.

Please note that not performing semantic integration of data sources also has a drawback. For instance, functional annotation of proteins in COLUMBA can be found both in SWISS-PROT description lines as well as in GO annotations. However, we strongly believe that integrating such data is counter-productive. Biologists heavily depend on the credibility of a source and the specific process-of-generation for judging data from databases. Therefore, our users have a clear separation between functional annotation in SWISS-PROT (developed and maintained in human language by a human expert) and functional annotation in GO (also maintained by humans, but in the form of IDs referencing external keywords). Integrating such information would not only be a tremendous and extremely expensive task, but would also blur this vital difference and thus leave users with less confidence to the system.

4.2.1 System Maintenance and Development

Focusing on individual data sources and the distinction between horizontal and vertical integration flows results in a modular system, benefiting extensibility and software development. Adding additional sources or removing existing ones is easy because in case of additions we extend the global schema by including the target schema of the new source as an additional dimension and in case of removal just delete the target schema and the linking tables from the global schema. The modular design also reduces the effort necessary for reacting on changes in the schemas of individual sources. Here again, we only take the individual source into account, define mappings and linkages for them, while the rest of the schema and database remains untouched.

The strong focus on individual data sources also has an advantage in creating the system software for loading and updating the database because in many cases we are able to apply existing software for parsing and loading (see Section 3). Specifying and implementing software parsers is also much simpler if we do not have to take into account any semantic mappings and global constraints. This accelerates the software development part and keeps the resulting system maintainable and well structured.

4.2.2 Data Provenance and Currency

Integrating the data in primarily separate parts for each source facilitates the determination of data provenance, i.e., reconstructing for each piece of information the source and data item it originates. We store information about data provenance and the data integration process as meta-data within separated tables in the global schema. This includes the original data source, the release number, and the programs used to extract or calculate the attribute values for each entry.

Detailed information about the data source of origin for an entry has advantages in case of error identification and source updates. If we identify a specific source entry to be erroneous we can correct it or simply exclude it from our database. In case of changes to source data we can identify those entries, which have to be updated or deleted within our database.

The mainly autonomous global schema design eases the update process in general because we upload the new release of a data source and are able to separately generate a new target sub-database, which is then connected to the existing data. Otherwise, there would be the necessity to merge the data from the new release with the existing and unchanged data from additional sources to generate a new version of the database.

4.2.3 Usability

In keeping data from the individual sources identifiable within the global schema the users can easily formulate the query on well known concepts. Schema integration over domains with a high degree of semantic heterogeneity like the life science or genome data domain result in an integrated schema which is rather complex (when using schema integration), making the schema difficult to understand. The resulting schema often differs heavily from the original schema, making it hard for the domain expert, used to the original sources, their concepts and terminology, to find their way around.

The multidimensional data integration approach also allows for a very intuitive query model. Queries are designed as refinements or filters over properties of proteins as derived from the different data sources. We expand on this aspect in Section 6.

5 Integration Process and Schema Design

5.1 COLUMBA Schema Design

The resulting schema (Fig. 2) is organized around PDB entries. They are surrounded by information from the additional sources, listed in Section 3, defining certain properties of the structures, contained within the entry. Each of the sources can be viewed as a dimension in the multidimensional model.

We want to stress again the importance of not merging data sources, although this might yield data redundancies within the integrated system. In many cases, meaningful schema integration is not possible because of the different concepts used for representing semantically equivalent information. For example SCOP and CATH, two databases storing classifications about evolutionary and structurally related proteins, use different classification hierarchies and methods resulting in different hierarchies despite many sub-trees of the classification being congruent. These differences would inevitably result in conflicts when both data sources should be integrated to one target table. No matter how these conflicts would be solved, the result would be unrecognizable for at least one of the data sources. Thus, we decided to avoid this problem by storing both hierarchies separately and leave it to the query system and/or the end user to decide, which of the sources is to be used.

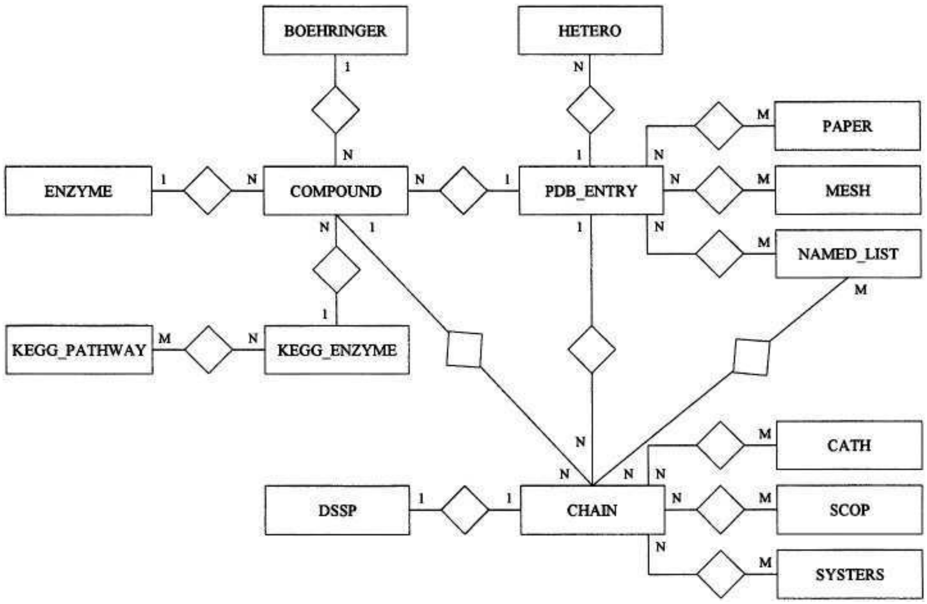


Fig. 2. Entity-Relationship model of COLUMBA. Data in the tables PDB_ENTRY, COMPOUND, CHAIN and HETERO originate from the PDB database, all others from the annotation data sources

5.2 COLUMBA Production Workflow

The production workflow (Fig. 3) of COLUMBA, like the schema, is also centered on PDB entries. Each of theses entries contains information about atom coordinates, macromolecules, their sequence, publications etc. PDB entries are downloaded in the mmCIF format and parsed into a relational schema using the OpenMMS toolkit [22] (Fig. 3.1). This schema is then mapped to a simpler representation (Fig. 3.2).

The annotation pipeline generates connections between PDB entries and objects from the other data sources (Fig. 3.3 / 3.4). Conceptually, each data source is represented by a software module, which implements a fixed interface. After having transformed a new PDB entry into the COLUMBA representation the workflow manager triggers each module by giving them the opportunity to detect and store annotations. Each data source module contains a list of data sources it requires. The workflow manager resolves these dependencies and finds a sequence of handling the modules. To include a new data source, only the tables and the module itself have to be written, and the module name added to a configuration file.

The modules vary according to the nature of the data source. For instance, the DSSP module calls the DSSP program and computes the secondary structure for each chain, whereas the SCOP module only takes the PDB and chain identifiers and matches them with classification files. The entire COLUMBA global schema contains 32 tables and is implemented on PostgreSQL 7.4.

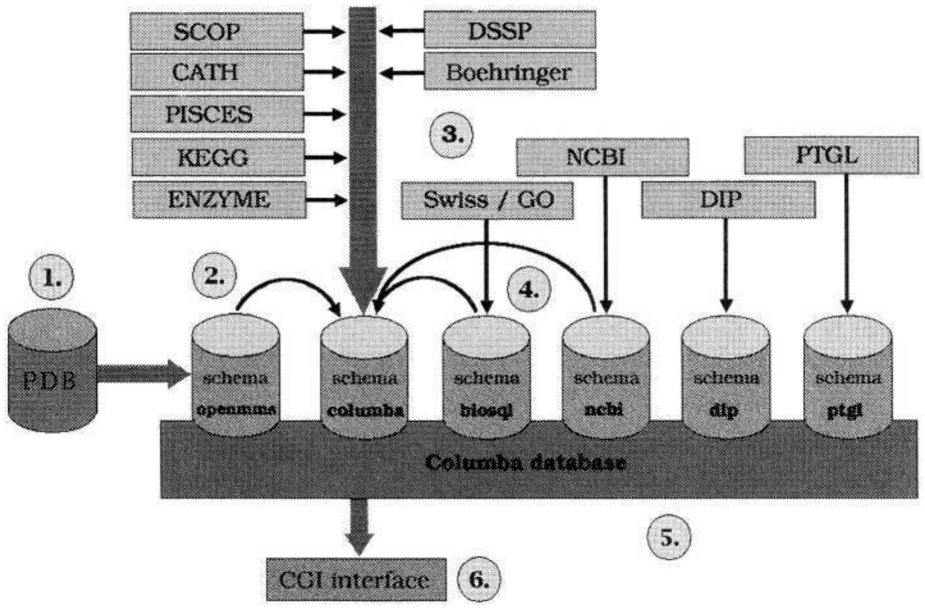


Fig. 3. Production workflow describing data fluxes and content relations: 1.) Load PDB data to OpenMMS database. 2.) Map OpenMMS schema to a simpler model. 3.) Add annotation from multiple sources of data. 4.) Add data to application-specific schemas. 5.) Have everything in one database. 6.) Query the data

5.3 Data Cleansing

The existence of errors and omissions in genome databases is a well known fact. The process of finding and resolving data quality problems in databases is generally called data cleansing [23]. We roughly differentiate between syntax and semantic errors. Syntax errors are mainly domain or format violations in data entries and values as well as misspellings. Syntactic cleansing operations like format and domain transformation, standardization, normalization, and dictionary lookup can be performed within the individual parsers or by separated cleansing modules for each of the sources. Currently, we are only checking for entry and value format violations and correct them by skipping the respective entry or value.

Semantic errors impact the accuracy of data regarding the real-world facts they are to represent. This type of errors is difficult to detect and eliminate. The direction we are currently following for semantic data cleansing utilizes redundant information. Exploiting redundancies for data cleansing is possible in cases where there exist overlapping versions of the same data source with slightly differing content. The differences might be due to manual data cleansing, data transformations, or data enhancement. The PDB database is available in three different versions and formats: (i) the original PDB data available in flat file format, (ii) data in macromolecular Crystallographic Information File format (mmCIF) from the PDB uniformity project at the University of California San Diego (UCSD) aiming at removing inconsistencies in PDB data [24], and (iii) the macromolecular structure relational database (E-MSD), a comprehensive cleansing project at the European Bioinformatics Institute (EBI) to ensure data uniformity and create a single access point for protein and nucleic acid structures and related information, available as Oracle dump files [25].

We currently utilize the parser for PDB flat-files to create an instance of our PDB target schema and the OpenMMS Toolkit, containing software for parsing and loading mmCIF files into a relational database. This toolkit uses a complex schema consisting of approximately 140 tables. We generated a set of schema mapping rules, which transform the data from the OpenMMS schema into a simpler target schema comprising only 6 tables. Thereby, we are able to create two overlapping instances for our PDB target schema. By comparing these instances and highlighting and evaluating the differences, using domain knowledge we are able to identify the reliable parts within both of the instances to select them for integration into a resulting third instance of the PDB target schema.

Identification of corresponding records within the two instances is easily done via the unique PDB identifier forming a matching record pair. We analyze these matching pairs for mismatches in their attribute values. The percentage of mismatches within the attributes varies widely (Table 2). Attributes having close to 100% mismatches often result from different formats or NULL values within one of the instances. Further investigating the mismatch causing values within each of the attributes reveals additional information about the causes for the mismatch.

For instance, comparison and evaluation enabled us to identify 32 records having a deposition year of *1900* in the mmCIF files where the original PDB flat files state the year *2000* for entry deposition. In an other case the structure method for over 2000 of the records resulting from parsing the PDB flat files was *unknown* while the mmCIF files stated *X-ray diffraction* as the structure method used.

We are also investigating mismatch dependencies between attribute pairs (A_i, A_j) where a mismatch in attribute A_i always causes additional mismatches in attribute A_j within the same matching record pair. Identification of these dependencies gives us a hint of the cause of the problem and hence of possible resolution strategies.

We are planning on extending this comparison approach to include E-MSD, giving us three overlapping instances at hand and also on extending the search for patterns of differences within the instances to gain a resulting integrated instance of even higher accuracy for PDB data.

Table 2. Showing for each of the attributes in table PDB_ENTRY the probability P(A_i) for mismatches within matching pairs and the number of different values for each attribute in each of the compared instances PDB-Parsed (resulting from PDB flat files) and MMS (resulting from OpenMMS mapping)

Attribute	P(A_i)	PDB-Parsed	MMS
NAME	0.999	2,299	19,736
YEAR_PDB_DEPOSITION	0.004	33	34
DEPOSITION_DATE	0.006	3,755	3,949
RELEASE_DATE	0.572	1,320	1,184
STRUCTURE_METHOD	0.183	135	96
RESOLUTION	0.451	289	356
R_VALUE	0.999	1	851
R_FREE	0.999	1	1,492
REFINEMENT_PROGRAM	1	1	485

6 Web Interface

For accessing COLUMBA a web interface is available at www.columba-db.de. This web interface uses a "query refinement" paradigm to return a subset of the PDB entries. A query is defined by entering restriction conditions on the data source specific annotations (Fig. 4). The user can combine several queries acting as filters to obtain the desired subset of PDB entries. The interface supports interactive and exploratory usage by straightforward adding, deleting, restricting or easing of conditions. For example, the whole set of PDB entries first can be filtered by searching for a name, then a constraint on the resolution of the structures is applied, and finally redundant sequences are excluded. The user is supported by a preview, which constantly shows the number of PDB entries and chains in the result set.

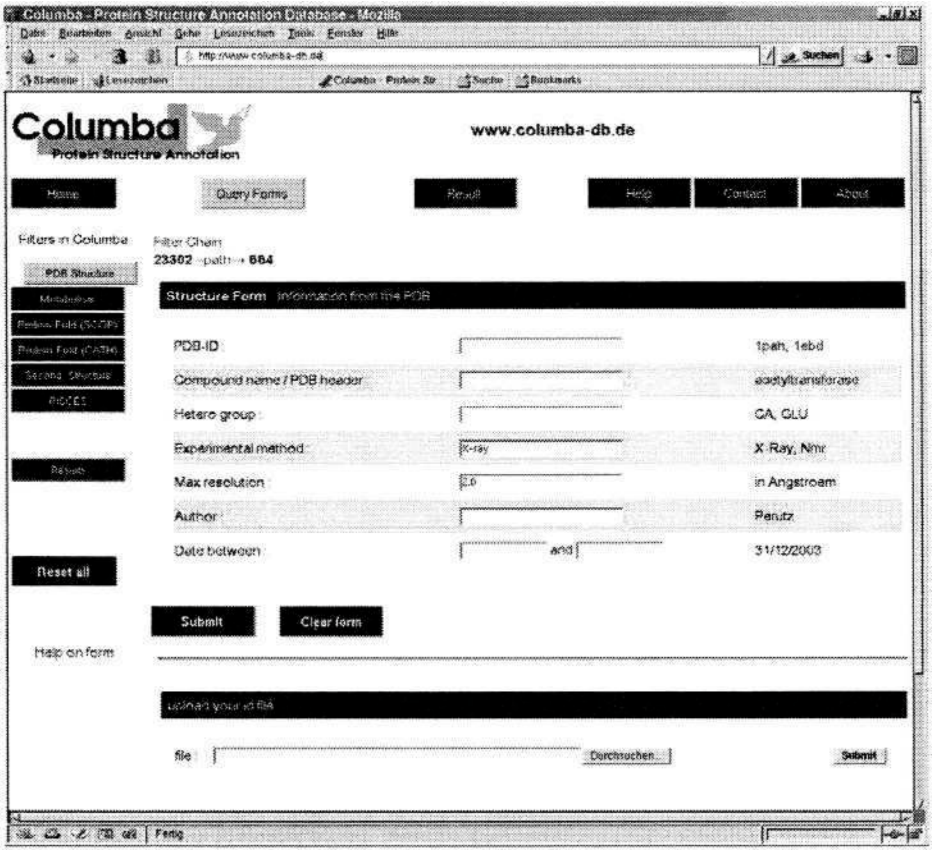
The result set gives basic information to each of the entries returned. This set is available as a formatted HTML table or text file. The user can see the full scope of COLUMBA on a single PDB entry, where all the information from the different data sources is shown.

7 Related Work

There are many efforts providing data on proteins via the Internet, and most of them use relational databases. Like the data sources listed above most of them focus on one special aspect of proteins, thus being an interesting object for integration. Often, they also contain hyperlinks to related databases, following the paradigm of link integration [26].

The PDBSUM [27] service is an excellent example for this approach: It contains a quick summary page for each PDB entry, plus dozens of links. The PDBSUM example is an auxiliary source for gathering information on few structures with known PDB codes, but it is not intended as a query and cross-comparison system.

The Jena Image library [28] provides data on protein structures and some external data sources (NDB, SWISS-PROT) in the form of large ASCII dump files. This approach has an intuitive charm for computer scientists, because parsing does not require much more than to split a string by tabs, but biologists will be deterred. Knowing this, the Jena team has designed a query interface, allowing full text search on all entries.



Filter Chain
23302 -path→ 664

Structure Form information from the PDB

PDB-ID		1pah, 1ebd
Compound name / PDB header		acetyltransferase
Hetero group		CA, GLU
Experimental method	X-ray	X-Ray, Nmr
Max resolution	2.0	in Angstroem
Author		Perutz
Date between	and	31/12/2003

upload your id file

file

Fig. 4. The COLUMBA Query Interface for specifying restrictions on attributes within the PDB target schema

Finally, there are several integrated databases of protein sequence and genome annotation (e.g. EXPASY [29]). They have been present for years, resulting in mature and reliable web interfaces and tools, which are very frequently used by lab biologists. Although they all contain references to PDB structures, they do neither cover the full spectrum of structure-specific data sources nor the level of detail contained in COLUMBA. The reason for this is obvious: The protein sequence databases are more than ten times larger than the PDB, shifting the focus from the mentioned databases to other fields.

In [30] a data warehouse approach to microarray data is described and also the integration of gene function and gene products is reported. However, no particular integration method is described, which makes a comparison impossible.

The various approaches for schema integration have already been mentioned and advantages and drawbacks discussed. Regarding schema design [31] suggests conceptual models for different types of Life Science data. However, these models are not designed for data integration and no model is proposed for protein structures.

Existing approaches for data cleansing are surveyed in [32]. These approaches currently focus predominantly on syntactical data cleansing while the aspect of semantic data cleansing remains unattended to a great extent. At the moment semantic data cleansing is mostly done using integrity constraint checking for error detection while error correction is often left up to the user. Exploiting overlapping information sources for data cleansing has not been regarded so far.

8 Conclusion

We presented COLUMBA, a database of protein annotations, which currently integrates data from seven data sources including PDB. We described COLUMBA's general architecture, sketched the advantages of the multidimensional approach to data integration implemented in COLUMBA, and discussed first results on data quality estimations we investigate. COLUMBA is freely available on the web for non-commercial and academic usage. Commercial users must obey all license restrictions of each individual data source integrated into COLUMBA. The database is already used intensively in several projects within the Berlin Center for Genome-based Bioinformatics (BCB), such as for the study of DNA-binding proteins within the department for biochemistry of the Charité and for the selection of candidate structures for the evaluation of parallel threading algorithms at the Conrad-Zuse Center.

Acknowledgement

This work is supported by BMBF grant no. 0312705B (Berlin Center for Genome-Based Bioinformatics) and DFG grant number GRK 316 (Berlin-Brandenburg School on Distributed Information Systems). We thank Raphael Bauer, Rene Heek, and Stephan Guenther for implementing many parts of COLUMBA. We truly acknowledge the excellent work of the database maintainer of the different source databases and thank for their consent to integrate their data into COLUMBA.

References

1. Bernstein, F.C., Koetzle, T.F., Williams, G.J.B., Meyer, E.F. Jr., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanouchi, T., Tasumi, M.: The Protein Data Bank: a computer-based archival file for macromolecular structures. J. Mol. Biol., Vol. 112 (1977) 535-542

2. Karp, P. D.: What we do not know about sequence analysis and sequence databases. Bioinformatics 14(9) (1998) 753-754
3. Devos, D. and Valencia, A.: Intrinsic errors in genome annotation. Trends in Genetics 17(8) (2001)429-431
4. Gilks, W.R., Audit, B., De Angelis, D., Tsoka, S., Ouzounis, C.A.: Modeling the percolation of annotation erros in a database of protein sequences. Bioinformatics, Vol. 18(12) (2002) 1641-1649
5. Kanehisa, M., Goto, S., Kawashima, S., Nakaya, A. The KEGG database at GenomeNet. Nucleic Acid Research, Vol. 30(1) (2002) 42-46
6. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol. Vol. 247 (1995) 536-540
7. Kabsch, W., Sander, C.: Dictionary of protein secondary structure: pattern of recognition of hydrogen-bonded and geometrical features. Biopolymers, Vol. 22(12) (1983) 2577-2637
8. Bairoch, A.: The ENZYME database. Nucleic Acid Research, Vol. 28(1) (2000) 304-305
9. Preissner, R., Goede, R., Froemmel, C.: Dictionary of interfaces in proteins (DIP). Databank of complementary molecular surface patches. J. Mol. Biol., Vol. 280, No. 3 (1998) 535-550
10. Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M.: CATH- A Hierarchic Classification of Protein Domain Structures. Structure, Vol. 5(8) (1997) 1093-1108
11. Boeckmann B., Bairoch A., Apweiler R., Blatter M., Estreicher A., Gasteiger E., Martin M. J., Michoud K., O'Donovan C., Phan I., Pilbout S., Schneider M.: The Swiss-Prot protein knowledgebase and its supplement TrEMBL. Nucleic Acids Research, Vol. 31(1) (2003) 365-370
12. Wheeler, D.L., Church, D.M., Federhen, S., Lash, A.E., Madden, T.L., Pontius, J.U., Schuler, G.D., Schriml, L.M., Sequeira, E., Tatusova, T.A., Wagner, L.: Database resources of the National Center for Biotechnology. Nucleic Acids Res., Vol. 31(1) (2003) 28-33
13. Ashburner M., Ball C.A., Blake J.A., Botstein D., Butler H., Cherry J.M., Davis A.P., Dolinski K., Dwight S.S., Eppig J.T., Harris M.A., Hill D.P., Issel-Tarver L., Kasarskis A., Lewis S., Matese J.C., Richardson J.E., Ringwald M., Rubin G.M., Sherlock G.: Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature Genetics 25(1) (2000) 25-29
14. Wang, G., Dunbrack, R.L. Jr.: PISCES: a protein sequence culling server. Bioinformatics, Vol. 19(12) (2003) 1589-1591
15. Krause, A., Stoye, J., Vingron, M.: The SYSTERS protein sequence cluster set. Nucleic Acids Res., Vol. 28(1) (2000) 270-272
16. Michal, G.: Biochemical Pathways. Boehringer Mannheim GmbH (1993)
17. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Wheeler, D.L.: GenBank. Nucleic Acids Res., Vol. 31(1) (2003) 23-37
18. Lakshmanan, L., Sadri, F., Subramanian, I.: On the Logical Foundation of Schema Integration and Evolution in Heterogeneous Database Systems. Intl. Conference on Deductive and Object-Oriented Databases (DOOD) (1993) 81-100
19. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. Conference on Very Large Data Bases(VLDB) (2002) 610-621
20. Rahm E., Bernstein P. A.: A survey of approaches to automatic schema matching. VLDB Journal, Vol. 10 (2001) 334-350
21. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record Vol. 26 (1997) 65-74
22. Greer, D.S., Westbrook, J.D., Bourne, P.E.: An ontology driven architecture for derived representations of macromolecular structure. Bioinformatics, Vol. 18(9) (2002) 1280-1
23. Rahm, E., Do, H.H.: Data Cleaning: Problems and current approaches. IEEE Bulletin of the Technical Committee on Data Engineering, 23(4) (2000)

24. Bhat, T.N., Bourne, P., Feng, Z., Gilliland, G., Jain, S., Ravichandran, V., Scheider, B., Schneider, K., Thanki, N., Weissig, H., Westbrook, J., Berman, H.M.: The PDB data uniformity project, Nucleic Acid Research, Vol. 29(1) (2001) 214-218
25. Boutselakis, H., Dimitropoulos, D., Fillon, J., Golovin, A., Henrick, K., Hussain, A., Ionides, J., John, M., Keller, P.A., Krissinel, E., McNeil, P., Naim, A., Newman, R., Oldfield, T., Pineda, J., Rachedi, A., Copeland, J., Sitnov, A., Sobhany, S., Suarez-Urunea, A., Swaminathan, J., Tagari, M., Tate, J., Tromm, S., Velankar, S., Vranken, W.; E-MSD: the European Bioinformatics Institute Macromolecular Structure Database. Nucleic Acid Research, Vol. 31(1) (2003) 458-462
26. Stein, L.: Creating a bioinformatics nation. Nature, Vol. 417(6885) (2002) 119-120
27. Laskowski, R.A.: PDBsum: summaries and analyses of PDB structures. Nucleic Acids Research, Vol. 29(1) (2001) 221-222
28. Reichert, J., Suhnel, J.: The IMB Jena Image Library of Biological Macromolecules: 2002 update. Nucleic Acids Res., Vol. 30(1) (2002) 253-254
29 Gasteiger E, Gattiker A, Hoogland C, Ivanyi I, Appel RD, Bairoch A.: ExPASy: The proteomics server for in-depth protein knowledge and analysis. Nucleic Acids Research, Vol. 31(13) (2003) 3784-3788
30. Cornell, M., Paton, N. W., Shengli, W., Goble, C. A., Miller, C. J., Kirby, P., Eilbeck, K., Brass, A., Hayes, A. and Oliver, S. G.: GIMS - A Data Warehouse for Storage and Analysis of Genome Sequence and Function Data. 2nd IEEE International Symposium on Bioinformatics and Bioengineering, Bethesda, Maryland (2001)
31. Paton, N. W., Khan, S. A., Hayes, A., Moussouni, F., Brass, A., Eilbeck, K., Goble, C. A., Hubbard, S. J. and Oliver, S. G.: Conceptual Modelling of Genomic Information. Bioinformatics 16(6) (2000) 548-557
32. Müller, H., Freytag, J.C.: Problems, Methods, and Challenges in Comprehensive Data Cleansing. Technical Report HUB-IB-164, Humboldt University Berlin, (2003)

On the Integration of a Large Number of Life Science Web Databases*

Zina Ben Miled[1], Nianhua Li[2], Yang Liu[1], Yue He[1], Eric Lynch[2], and Omran Bukhres[2]

[1] Department of Electrical and Computer Engineering, Indianapolis University Purdue University Indianapolis, Indianapolis IN 46202, USA,
zmiled@engr.iupui.edu
[2] Department of Computer and Information Science, Indianapolis University Purdue University Indianapolis, Indianapolis IN 46202, USA

Abstract. The integration of life science web databases is an important research subject that has an impact on the rate at which new biological discoveries are made. However, addressing the interoperability of life science databases presents serious challenges, particularly when the databases are accessed through their web interfaces. Some of these challenges include the fact that life science databases are numerous and their access interface may change often. This paper proposes techniques that take into account these challenges and shows how these techniques were implemented in the context of BACIIS, a federation of life science web databases.

1 Introduction

In the past decade the management and interoperability of life science databases has been an active research area due to an increase in the volume and the complexity of biological data. The design and development of new sequencing and high throughput screening technology resulted in a large number of scientists contributing data to various public domain databases. The challenge of the next decade is to develop new technologies that can facilitate the management of this data in order to allow scientists to extract new knowledge, which can accelerate research findings in functional genomics, proteomics, and drug discovery.

Several commercial and non-commercial integration systems for life science databases have been proposed [1,2,3,4,5,6]. These systems use a wide range of underlying architectures. For example, NCBI/Entrez [5] uses the link driven approach to integrate various databases. GUS [3] uses a data warehouse approach. TAMBIS [7] is a database federation that is based on a mediator-wrapper architecture. The role of the wrapper is to extract information from remote databases, and the role of the mediator is to integrate the data. In addition to the integration approach, some of the previously proposed systems target web databases

* This work is supported in part by NSF CAREER *DBI-0133946* and NSF *DBI-0110854*

directly in an effort to preserve their autonomy. Therefore, no assumption is made as to the active participation of the remote databases. Examples of these systems include TAMBIS and BACIIS [6]. Other previously proposed systems assume that either the schema of the remote databases is known or simply that the data is distributed by the remote database on a regular basis in a text based format that can be imported into a local database. Examples of these systems include Discovery Link [4] and SRS [8]. Although the integration of life science databases has been well studied in the literature, there are several remaining important research issues that need to be addressed. Given that the number of life science databases is currently more than 400 [15] and it keeps growing, any proposed architecture to the integration of these databases should be scalable. Furthermore, the integration system should easily adapt to changes in the web interfaces of the remote databases. This latter aspect is particularly important if the integration system accesses the remote databases only through their web interface. In general, a wrapper is needed to extract information from each of the remote databases. SRS supports 72 wrappers and there are more than 500 wrappers in the public domain for SRS [9]. K1 [10] has about 60 to 70 types of wrappers. TAMBIS uses more than 300 CPL functions defined by BioKleisli [11]. Each CPL function is a wrapper function which extracts a single type of data from a single data source [12]. These wrappers can easily fail when the interfaces of the corresponding databases change. In this paper we focus on the issues of supporting the integration of a large number of web databases when the remote databases are only accessible through their often changing web interfaces. The proposed techniques are implemented in BACIIS (Biological and Chemical Information Integration System) [6,13]. These techniques include the support for the transparency of the remote databases, the separation of the global schema from the data source description and the support for wrapper induction. Section 2 of this paper presents an overview of the BACIIS architecture. The domain ontology, query planner, and data source wrapper are described in sections 3, 4 and 5 respectively. Section 6 summarizes the contributions of this paper.

2 BACIIS Architecture

BACIIS is an ontology-centered information integration system for life science web databases. It supports the seamless semantic-based integration of multiple heterogeneous life science web databases and allows the execution of global applications that extend beyond the boundaries of individual databases. The architecture of BACIIS (Figure 1) uses a widely adopted mediator-wrapper approach. The wrapper layer extracts information from the remote data sources. Source specific data retrieval rules, such as site URLs and data field labels, are defined in a set of data source schema files in XML formats. The schema file serves as a knowledge base for a generic wrapper engine in order to form fully functional wrappers. The separation of the source specific information from the generic data retrieval functions facilitates wrapper construction and maintenance as discussed in Section 5. The mediator layer resolves data interoperability among data sources. Its core consists of a domain ontology which represents the biological knowledge domain. This ontology serves as global schema. Mapping relations are defined between the data source schema files and the domain ontology. Data retrieved from the remote sources is labeled with ontology concepts. This allows data labeled with the same ontology concepts to easily be integrated. As mentioned above, data extraction rules for individual data sources are specified in the data source schema files. The separation of source specific information from domain ontology improves the stability of the domain ontology and the scalability of BACIIS in terms of adding new databases. This aspect is discussed further in Section 3. In BACIIS, the data sources are transparent to the user. The queries are expressed by using database independent bioinformatics terms. It is the responsibility of the query planner in BACIIS to discover related data sources and extract information from them. This design decision allows the scientist to extract information from a large number of databases without being familiar with any of the databases. Furthermore, new databases can be added to BACIIS without any change to the query submission method.

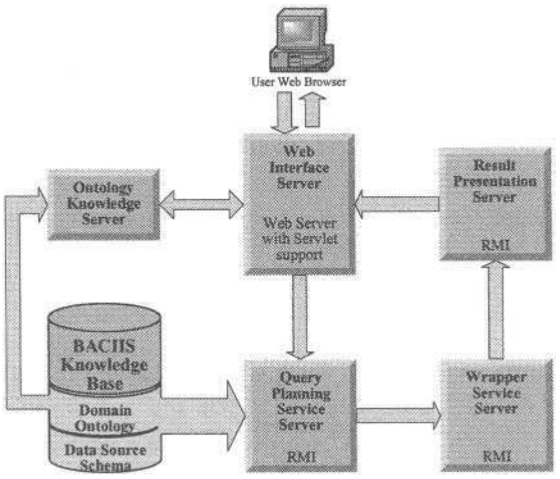


Fig. 1. BACIIS System Architecture

3 Domain Ontology

The BACIIS Ontology (BaO) [13] aims at representing the biological domain knowledge rather than accommodating the schema of all the data sources. As mentioned in Section 2, BACIIS has a schema file for each life science data source. This file is used to specify the query interface and the result data structure of the corresponding data source. It is different from the real data source schema of the local database. In the schema file, data retrieval rules are labeled with domain ontology concepts rather than physical data organizations and relations. Therefore, the retrieved data can be reorganized according to the domain ontology, and then be semantically mapped onto the knowledge domain represented by BaO. From this point of view, BaO also serves as the global schema of BACIIS. The integration is performed in two steps in BACIIS: first a source-specific schema file defines the data retrieval rules and maps them to ontology concepts, and then the domain ontology defines the relationship among the ontology concepts. Mapping relations among different sources are established by sharing equivalent or related ontology concepts. Alternatively, one can attempt to integrate all data source specifications in one schema. However, as more data source are added, this approach becomes complex, rendering the maintenance of the integrated system impractical.

The global schema of BACIIS (BaO) is independent of the individual life science data source and will only change when the biological domain evolves to include new discoveries and concepts. The stability of the concepts in BaO promotes scalability in BACIIS because the addition of new data sources and updates in the data sources will not require changes in the ontology.

BaO consists of three top classes: *Object, Property,* and *Relation* as shown in Figure 2. The sub-classes of *Object* and *Property* are arranged into hierarchical trees by using the relationship *is-a-subset-of.* The relationship between the class of *Object* and *Property* is defined as *has-property.* These two types of relations (*is-a-subset-of* and *has-property*) are the main relations in BaO. Other relations such as *source-of, mark-on,* and *encode* are defined between the sub-classes of *Object.* The design of separate hierarchical structures for *Object, Property,* and *Relation* preserves the isolation of the classes in BaO [13], which reduces the maintenance effort needed to keep the ontology up-to-date. That is, changes to one part of the ontology, which represents a set of concepts, have limited impact on other parts of the ontology.

4 Query Planning

BACIIS uses the query-by-example approach to simplify the entry of complex queries such as "*What is the 3D structure of all proteins that belong to the enzyme family EC:1.1.1.1 and is located within the human chromosome region 4q21-4q23?*". Query formulation is guided by the ontology. The above query specifies a set of proteins by two constraints (i.e., *Cytogenetic-Region = 4q21-4q23* and *EC Number =1.1.1.1*), and then requires the information related to some aspects (i.e., *3D structure*) *of* the selected proteins. No individual life science data source can answer the above query directly due to limited query capabilities of the data sources [15]. For example, although PDB can provide protein *3D structure* information, it cannot identify a protein by the *cytogenetic region* of its encoding gene. BACIIS identifies query paths automatically. The data source schema files map source specific information to the ontology concepts. If two data sources have data fields mapped to the same ontology concept, the common concept serves as a link. By analyzing the data source schema, BACIIS discovers graph

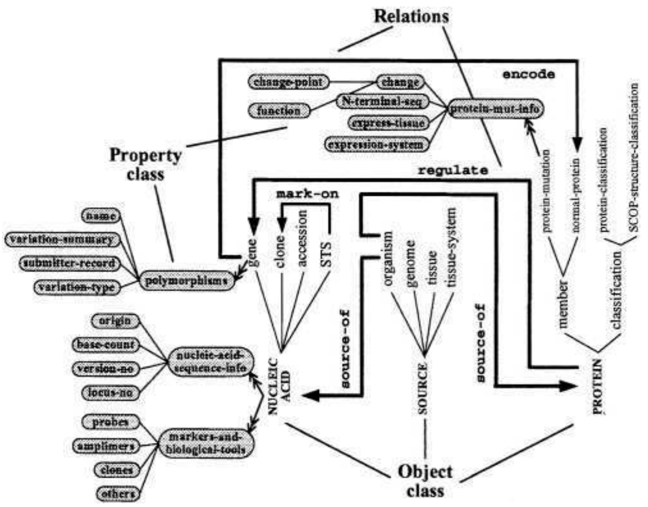


Fig. 2. The Structure of the ontology (BaO) in BACIIS

relating the concepts underlying every user query, and identifies the most efficient plan path. For example, all the possible paths from the input constraints (i.e., *Cytogenetic-Region = 4q21-4q23* and *EC Number =1.1.1.1*) to one data source of protein *3D-structure* (PDB) are shown in Figure 3. Paths are labeled A1 through A4 for the first input constraint (i.e. *Cytogenetic-region=4q21-4q23*) and B1 through B3 for the second input constraint (i.e. *Ec-No=1.1.1.1*). The query result is the intersection of the two selected paths. When the output information is available in more than one data source, one path will be identified for each source to maximize the answer completeness.

The data source transparency provided by BACIIS allows obtaining information from various data sources without knowing their query interfaces and schema. Furthermore, when BACIIS scales up to integrate more data sources, the users can get up-to-date information without any changes in the query submission.

5 Wrapper and Wrapper Induction

Wrappers are needed to extract the relevant data from the remote data sources. Creating a wrapper for a web data sources is a challenging task because first, most data sources are meant to be human readable rather than machine-readable, and second, web data sources are highly heterogeneous. Therefore, integration systems often use a set of data source specific wrapper modules or wrapper functions for data retrieval. The development of new wrappers is required when new data sources are included. This is a tedious process that hinders the scalability of the integration system. One important feature of BACIIS is that it separates

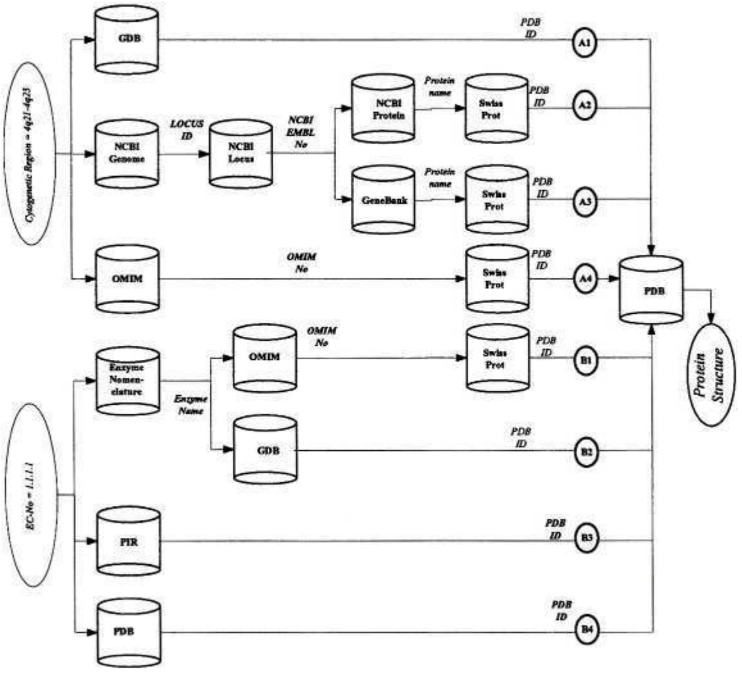


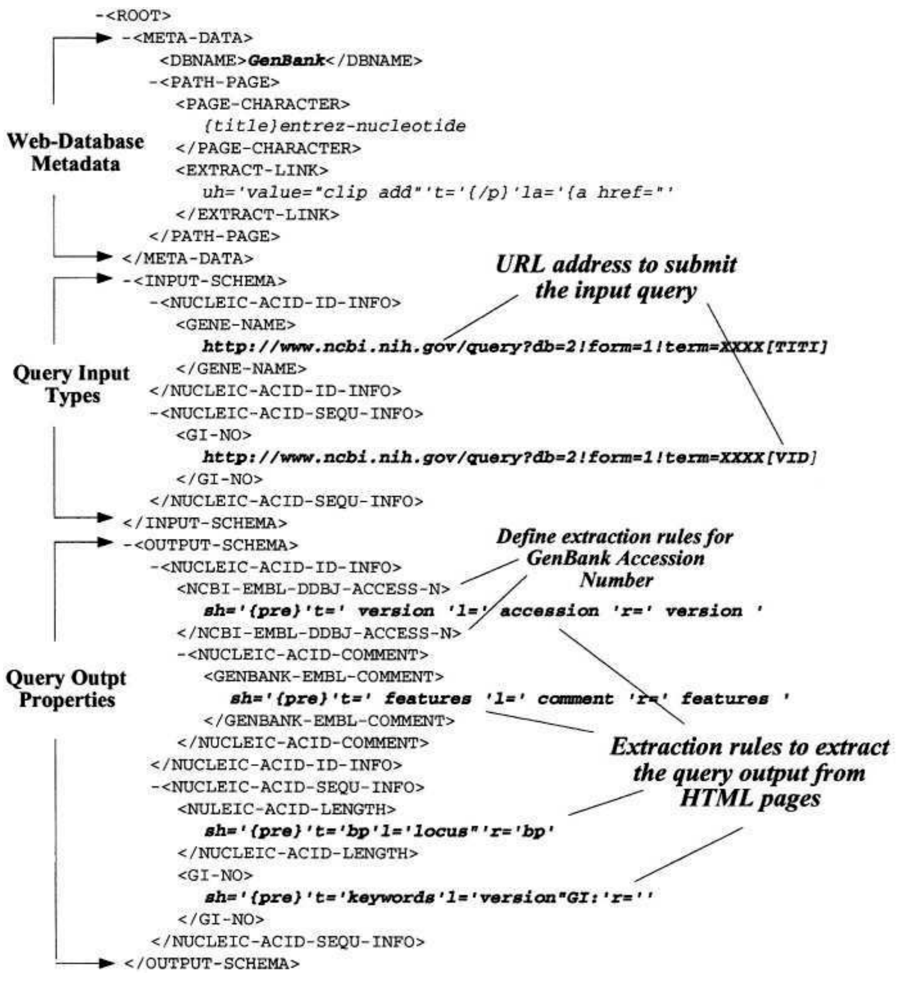
Fig. 3. All Possible Query Paths for the Sample Query

data source specific information from the wrapper executable module. A generic wrapper engine is used to extract information from all data sources. This engine uses the data extraction methods specified in the data source schema file of the individual data source. The separation between the wrapper engine and the data source schema promotes scalability by eliminating the need to change the source code of the wrapper engine when a web database is added to the system or when the interface corresponding to an existing database changes.

5.1 Data Source Schema

Each web database integrated in BACIIS is described by a data source schema file. Currently, BACIIS integrates GenBank [16], SWISS-PROT [17], PIR [18], PROSITE [19], ENZYME [20], PDB [21], and OMIM [22]. Figure 4 shows the data source schema of the database GenBank as an example.

For each life science web database, all the data types that can be submitted as query input and can be retrieved as query output from the web interface of the database are stored in the data source schema by using the concepts and relations defined in the ontology (BaO). For the INPUT-SCHEMA section shown in Figure 4, each XML tag name corresponds to a BaO property that can be accepted as input by a GenBank query. The tag structure follows the property



```
-<ROOT>
  -<META-DATA>
     <DBNAME>GenBank</DBNAME>
    -<PATH-PAGE>
       <PAGE-CHARACTER>
         {title}entrez-nucleotide
       </PAGE-CHARACTER>
       <EXTRACT-LINK>
         uh='value="clip add"'t='{/p}'la='{a href="'
       </EXTRACT-LINK>
     </PATH-PAGE>
  </META-DATA>
  -<INPUT-SCHEMA>
    -<NUCLEIC-ACID-ID-INFO>
       <GENE-NAME>
         http://www.ncbi.nih.gov/query?db=2!form=1!term=XXXX[TITI]
       </GENE-NAME>
     </NUCLEIC-ACID-ID-INFO>
    -<NUCLEIC-ACID-SEQU-INFO>
       <GI-NO>
         http://www.ncbi.nih.gov/query?db=2!form=1!term=XXXX[VID]
       </GI-NO>
     </NUCLEIC-ACID-SEQU-INFO>
  </INPUT-SCHEMA>
  -<OUTPUT-SCHEMA>
    -<NUCLEIC-ACID-ID-INFO>
       <NCBI-EMBL-DDBJ-ACCESS-N>
         sh='{pre}'t=' version 'l=' accession 'r=' version '
       </NCBI-EMBL-DDBJ-ACCESS-N>
      -<NUCLEIC-ACID-COMMENT>
         <GENBANK-EMBL-COMMENT>
           sh='{pre}'t=' features 'l=' comment 'r=' features '
         </GENBANK-EMBL-COMMENT>
       </NUCLEIC-ACID-COMMENT>
     </NUCLEIC-ACID-ID-INFO>
    -<NUCLEIC-ACID-SEQU-INFO>
       <NULEIC-ACID-LENGTH>
         sh='{pre}'t='bp'l='locus"'r='bp'
       </NUCLEIC-ACID-LENGTH>
       <GI-NO>
         sh='{pre}'t='keywords'l='version"GI:'r=''
       </GI-NO>
     </NUCLEIC-ACID-SEQU-INFO>
  </OUTPUT-SCHEMA>
```

Web-Database Metadata
URL address to submit the input query
Query Input Types
Define extraction rules for GenBank Accession Number
Query Outpt Properties
Extraction rules to extract the query output from HTML pages

Fig. 4. Data Source Schema for the Database GenBank Written in XML

hierarchy defined in BaO. The tag value is the URL address that is used by the wrapper to submit the query. The OUPUT-SCHEMA section also uses the BaO properties as tag names to represent the information that can be retrieved from GenBank. The tag values in the OUPUT-SCHEMA section correspond to the extraction rules that can be used by the wrapper to extract the required data from the HTML pages returned by the web database. The METADATA section contains the data name and the rules for auto-direction and identification of the final result page.

5.2 Wrapper Engine

The genetic wrapper engine of BACIIS is designed to submit queries and extract results from any web database participating in the integration. It consists of a Plan Parser, a Result Collector, and a Wrapper Engine as shown in Figure 5. The query plans are composed of one or more query execution paths. The Plan Parser parses the query plans and spawns multiple Wrapper Engine instances, with each corresponding to one path. Each plan path consists of one or more steps whose execution order is coordinated by the PlanStep Iterator in Figure 5. For each step, the Identification Module will first create an empty result container for the result data of this step. The result container is organized according to the BaO property hierarchy for the output property being retrieved by the corresponding step or sub-query. The Execution Module will use the query-specific URL and the input data set given by the PlanStep Iterator to compose the complete query URL address and use this URL to submit the sub-query on the remote web database. The returned HTML pages are forwarded to the Extraction Module together with the extraction rules. The Extraction Module extracts the result data from the returned pages and places them into the result container created by the Identification Module. The loaded result container will be transferred to the Temporary Data Pool in the wrapper engine. Depending on whether the return data is a final result data or an intermediate result data, the result data is sent to the Result Collector or feedback to the PlanStep Iterator, which will use it in the execution of the next step of the plan path.

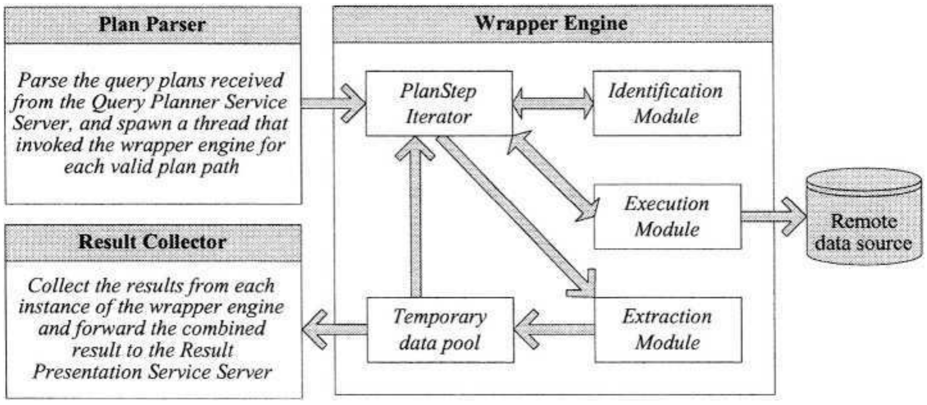
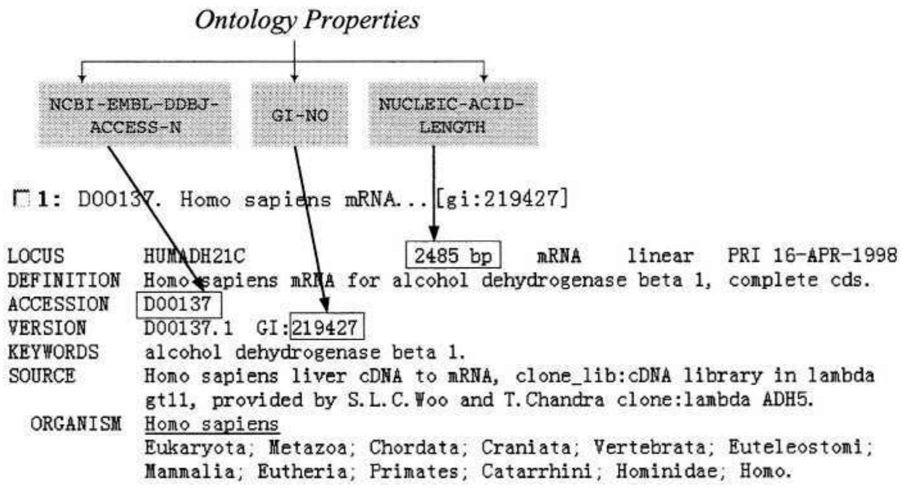

Plan Parser
Parse the query plans received from the Query Planner Service Server, and spawn a thread that invoked the wrapper engine for each valid plan path
Result Collector
Collect the results from each instance of the wrapper engine and forward the combined result to the Result Presentation Service Server
Wrapper Engine
PlanStep Iterator
Identification Module
Execution Module
Remote data source
Temporary data pool
Extraction Module

Fig. 5. The Architecture of BACIIS Wrapper Engine

5.3 Wrapper Induction

Wrapper induction refers to process of learning the extraction rules based on web page examples. It represents an attempt to overcome the tedious, time consuming, and error prone process associated with manually generating the extraction rules [23,24]. The approaches to wrapper induction proposed in the literature traditionally use machine learning [25,26], supervised learning [27], and inductive logic programming [28] techniques. In general, those techniques require a large number of hand labeled sample pages be fed to the induction system. In order to promote scalability, our focus was to develop a rapid and easy approach that does not depend on hand labeled examples and does not require a large number of sample pages. Furthermore, the extracted fields not only have to be identified, but they must also be labeled with ontology terms in order to facilitate the integration of the result returned from one web database with the result returned from another database. Therefore, the proposed wrapper induction algorithm has to be able to 1) identify the actual data that is to be extracted from the page, and 2) map the extracted data to the ontology terms.

Since all of the web pages returned from the web databases are meant to be human readable, the data in these pages is typically preceded by a label that identifies it. For example, as shown in Figure 6, when searching GenBank [16] for the accession number of a specific nucleotide, the result page will have the label " accession" associated with the accession number. In order to extract this number, the corresponding label has to be located in the page. The challenge is to first locate the label-data tuples and to associate these tuples with terms from the ontology.



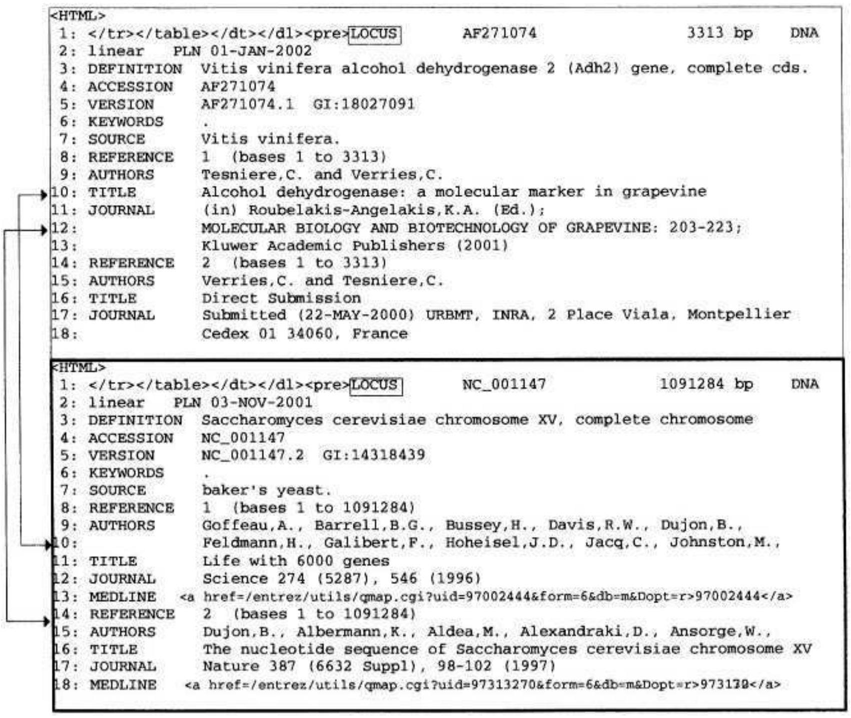
```
1: D00137. Homo sapiens mRNA... [gi:219427]

LOCUS       HUMADH21C            2485 bp    mRNA    linear   PRI 16-APR-1998
DEFINITION  Homo sapiens mRNA for alcohol dehydrogenase beta 1, complete cds.
ACCESSION   D00137
VERSION     D00137.1  GI:219427
KEYWORDS    alcohol dehydrogenase beta 1.
SOURCE      Homo sapiens liver cDNA to mRNA, clone_lib:cDNA library in lambda
            gt11, provided by S.L.C.Woo and T.Chandra clone:lambda ADH5.
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Primates; Catarrhini; Hominidae; Homo.
```

Fig. 6. Example Result Web Page from GenBank

Tuple Identification Tuple identification, in general, is a challenging research subject. Limiting the scope to a given domain such as the biological domain in the case of this paper helps reduce some of the difficulty, especially when an ontology that describes the domain is available. The proposed approach to tuple identification is an enhancement to the method for tuple identification introduced in [24]. It is based on discovering similarity and dissimilarity between sample pages from a website. The comparison process begins with the user supplying sample pages from the target life science web database. For the wrapper to be complete, the sample pages must be inclusive of all fields from the database. The tuple identification method was implemented and tested on the following biological databases: GenBank, SWISS-PROT, and PIR, and it was found that in general 3 to 9 sample pages were required for adequate tuple identification. The number of sample pages required varies with the quality of the provided sample pages.

The sample pages are parsed into lists of tokens consisting of either HTML tags or strings. The algorithm processes tokens from all of the lists simultaneously. The first sample is arbitrarily selected as the control page. The control page will have any missing fields added to it to construct a template with all available database fields. This control page will be presented to the user during the verification process. Tokens are compared across all the token lists in order, noting the similarities. Non-HTML tokens from the control list are checked against the knowledge base. If a token appears in the knowledge base, it is a possible hit on a label field. If the current token from a list does not match the current token in the control list, it too is checked against the knowledge base. This token may represent an optional field that is not present in the arbitrarily selected control list. As previously mentioned, this approach is similar to the approach proposed in [24]. The difference lies in the fact that the wrapper induction approach proposed in this paper does not need the HTML clues in the page in order to infer the structure of the HTML page. Some of the disadvantages of the approach proposed in [24] include the requirement that the sample pages be well structured, the need for a large number of sample pages, the difficulty in consolidating between mismatches, and the lack of a method that can identify a field once found. The approach proposed in this paper addresses these challenges by using the domain ontology. The advantages of the approach proposed in this paper for induction compared to the approach introduced in [24] are obtained only because the induction process is limited to a specific domain. The sample pages do not have to be well structured since the labels are expected to match terms in the ontology. Similarly, since labels are compared to ontology terms, only a representative set of pages is required. Furthermore, mismatches in the labels are resolved by checking if the corresponding labels are semantically equivalent in the ontology. Semantic variability in nomenclature is a very common phenomenon in the biological domain. In the ontology developed for BACIIS this semantic variability is resolved by identifying syntactically different terms that refer to the same semantic concept. For example, the database GenBank uses "REFERENCE", the database PDB uses "Primary Citation", and the database PIR uses "Bibliography" to refer to literature reference.

Figure 7 illustrates the proposed tuple identification approach using two partial pages from GenBank. This figure shows that HTML tags are not present in the data section of the page. GenBank, like many other Life Science databases, uses the < *pre* > tag to denote preformatted text. This characteristic of biological databases makes direct use of the approach proposed in [24] not possible, since it relies on well-structured pages and in particular on the presence of appropriate html tags for the labels.



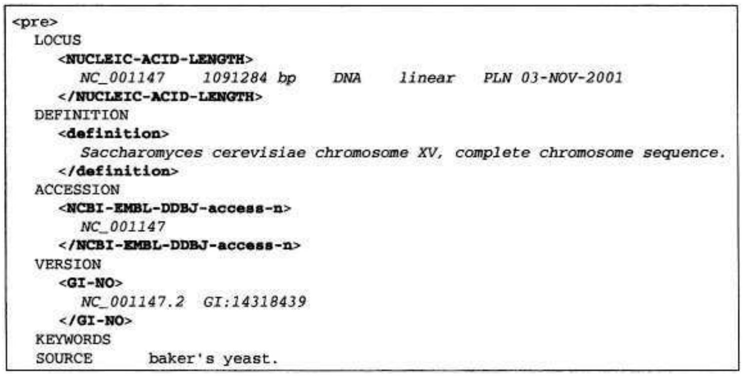
```
<HTML>
 1: </tr></table></dt></dl><pre>LOCUS       AF271074              3313 bp    DNA
 2: linear    PLN 01-JAN-2002
 3: DEFINITION  Vitis vinifera alcohol dehydrogenase 2 (Adh2) gene, complete cds.
 4: ACCESSION   AF271074
 5: VERSION     AF271074.1  GI:18027091
 6: KEYWORDS    .
 7: SOURCE      Vitis vinifera.
 8: REFERENCE   1  (bases 1 to 3313)
 9: AUTHORS     Tesniere,C. and Verries,C.
10: TITLE       Alcohol dehydrogenase: a molecular marker in grapevine
11: JOURNAL     (in) Roubelakis-Angelakis,K.A. (Ed.);
12:             MOLECULAR BIOLOGY AND BIOTECHNOLOGY OF GRAPEVINE: 203-223;
13:             Kluwer Academic Publishers (2001)
14: REFERENCE   2  (bases 1 to 3313)
15: AUTHORS     Verries,C. and Tesniere,C.
16: TITLE       Direct Submission
17: JOURNAL     Submitted (22-MAY-2000) URBMT, INRA, 2 Place Viala, Montpellier
18:             Cedex 01 34060, France
```

```
<HTML>
 1: </tr></table></dt></dl><pre>LOCUS       NC_001147             1091284 bp    DNA
 2: linear    PLN 03-NOV-2001
 3: DEFINITION  Saccharomyces cerevisiae chromosome XV, complete chromosome
 4: ACCESSION   NC_001147
 5: VERSION     NC_001147.2  GI:14318439
 6: KEYWORDS    .
 7: SOURCE      baker's yeast.
 8: REFERENCE   1  (bases 1 to 1091284)
 9: AUTHORS     Goffeau,A., Barrell,B.G., Bussey,H., Davis,R.W., Dujon,B.,
10:             Feldmann,H., Galibert,F., Hoheisel,J.D., Jacq,C., Johnston,M.,
11: TITLE       Life with 6000 genes
12: JOURNAL     Science 274 (5287), 546 (1996)
13: MEDLINE   <a href=/entrez/utils/qmap.cgi?uid=97002444&form=6&db=m&Dopt=r>97002444</a>
14: REFERENCE   2  (bases 1 to 1091284)
15: AUTHORS     Dujon,B., Albermann,K., Aldea,M., Alexandraki,D., Ansorge,W.,
16: TITLE       The nucleotide sequence of Saccharomyces cerevisiae chromosome XV
17: JOURNAL     Nature 387 (6632 Suppl), 98-102 (1997)
18: MEDLINE   <a href=/entrez/utils/qmap.cgi?uid=97313270&form=6&db=m&Dopt=r>973139</a>
```

Fig. 7. Example of the Label/Data Identification Algorithm Applied to GenBank

When the pages in Figure 7 are processed, the first label match is the token *LOCUS*. Both pages contain LOCUS and the term is also part of the ontology. The next tokens, AF271074 in (a) and NC_01147 in (b) do not match, so it is marked as the beginning of the data field. The tokens are then consumed and checked against the ontology until *DEFINITION* is found on line three. This is the next label hit since it exists in both pages and is an ontology term. Once this second label is identified, the algorithm backtracks one token and marks the end of the LOCUS data field. Figure 8 shows the control page after line 1 to line

7 of the lower part of Figure 7 have been processed and marked with ontology terms.



```
<pre>
  LOCUS
    <NUCLEIC-ACID-LENGTH>
      NC_001147     1091284 bp     DNA     linear    PLN 03-NOV-2001
    </NUCLEIC-ACID-LENGTH>
  DEFINITION
    <definition>
      Saccharomyces cerevisiae chromosome XV, complete chromosome sequence.
    </definition>
  ACCESSION
    <NCBI-EMBL-DDBJ-access-n>
      NC_001147
    </NCBI-EMBL-DDBJ-access-n>
  VERSION
    <GI-NO>
      NC_001147.2  GI:14318439
    </GI-NO>
  KEYWORDS
  SOURCE      baker's yeast.
```

Fig. 8. Partial Example Page after Label/Data Tuple Identification

The identification process continues until line 10 in Figure 8. This is a trivial mismatch caused by the variable length of the data field. Processing of the second list will pause on the term “title” until the first list reaches that point. Processing will then resume. The next mismatch is found between line 14 and line 13. One of these fields is an optional field that appears in at least one, but not all, of the pages. To proceed, the page missing the field is scanned ahead to ensure that “MEDLINE” does not occur later in the page. If it does, then “REFERENCE” could be the optional field. Once it is determined that “MEDLINE” is the optional field, the identification process is synchronized by using the label “REFERENCE”. The first list will then wait while the second list processes the fields up until “REFERENCE” at which point processing of both lists resumes.

There are several methods for representing extraction rules. These methods include HLRT and derivatives of this method such as LR, HBELRT and HOCLRT [29]. In this paper, the HLRT method, proposed in [29], was selected because it is expressive enough to accommodate biological web databases and it is easy to implement. This method represents extraction rules by defining the *head,* the *left,* the *right* and the *tail* delimiters. The *head* (H) and *tail* (T) are used to delineate the beginning and end, respectively, of the relevant information in a given web page. The *head* delimiter allows the wrapper to skip any superfluous information at the beginning of the page or to navigate past a particular landmark within a page. Each rule can specify a unique head as needed and the *head* delimiter can be used in conjunction with the *left* delimiter to form a discontinuous sequence of tokens that identify a field. The *tail* is useful particularly if optional labels occur at the end of the web page. The *left* (L) and *right* (R) delimiters are used repeatedly within a page to delineate the beginning and end, respectively, of data fields that correspond to labels.

The process begins by taking the set of sample pages that have been labeled by the label/data identification process. The algorithm locates the first ontology term marked in the sample and tries to determine the *head* and *left* delimiters. These two delimiters must be chosen together, as some choices for a *left* delimiter may only be valid when combined with a particular token as the *head.* In the example of Figure 6, the first ontology term is $< nucleic - add - length >$. All of the samples are examined and it is found that all $< nucleic - acid - length >$ tags are preceded by the label LOCUS. The $< pre >$ tag is also found as a unique tag that precedes the ontology tag. It is therefore chosen as a *head* delimiter. Once the *head* and *left* delimiters are found, the *end* delimiter is found and the *right* delimiter is sought. In all of the GenBank samples, DEFINITION follows the terminating tag and it is chosen as the *right* delimiter. DEFINITION can be selected as the *tail* as well. This results in the following extraction rule for $< nucleic - acid - length >$ ($< nucleic - acid - length >$: H= $< pre >$, L = LOCUS, R = DEFINITION, T = DEFINITION). This rule will be interpreted by the wrapper engine as: skip everything until the token $< pre >$ is found, then look for LOCUS. When it is found, begin extracting all data after LOCUS until DEFINITION is encountered.

Validation The wrapper induction system was tested by using three representative life science web databases that BACIIS currently integrates, namely GenBank, PIR and SWISS-PROT. Three to nine sample pages from each web database were selected. The extraction rules generated by the wrapper induction system were compared to the manually written rules for these databases. The extraction rules that were manually created are part of the BACIIS system and have been used to correctly answer queries that involve the selected databases.

The algorithm correctly identified the label and data fields of the desired fields for 16 of the 21 fields without any user intervention. This included correctly identifying the appropriate *head* and *tail* delimiters and the associated ontology terms for the field. While the induction algorithm was able to generate a wrapper to extract most of the data and to properly classify the labels using ontology terms, some fields were not correctly processed. For example, consider the "MEDLINE" field in Figure 7. In the manually written extraction rules the data field contains only the number 97002444, whereas the wrapper induction algorithm generated an extraction rule where the data field is the entire link address (line 13 of Figure 7). Future work will focus on developing methods to facilitate sub-data field extraction such as the one illustrated by this example.

6 Conclusions and Future Work

Life science databases have seen an important increase in number in the past decade due to intensive research in areas such genomics and proteomics. Several research studies cross the boundaries of individual databases and can therefore benefit from a system that integrates these databases. This paper investigates the issues of scalability when integrating a large number of autonomous life science web databases. The approaches proposed in this paper include 1) the separation of the domain ontology which is used as the global schema and the description of the data sources, 2) the separation of the data extraction rules which are included in the data source schema file and the wrapper engine 3) the development of a wrapper induction system which can generate the extraction rules for a give web database using a reduced number of untagged sample pages and 4) the development of a query planner that can identify the sources databases needed to answer a user query, thereby guarantying total transparency of the web databases. The techniques proposed in this paper were implemented in an integration system for life science web databases (BACIIS) which is available at `http://baciis.engr.iupui.edu`

References

1. Goble, C.A., Stevens, R., Ng, G., Bechhofer, S., Paton, N.W., Baker, P.G., Peim, M., Brass, A.: Transparent Access to Multiple Bioinformatics Information Sources. IBM Systems Journal. **40(2)** (2001) 532-552
2. Zdobnov, E. M., Lopez, R., Apweiler, R., Etzold, T.: The EBI SRS server-recent developments. Bioinformatics **18** (2002) 368-373
3. Davidson, S. B., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., Stoeckert, C.: K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources. IBM Systems Journal **40(2)** (2001)
4. Haas, L. M., Rice, J. E., Schwarz, P. M., Swops, W. C., Kodali, P., Kotlar, E.: DiscoveryLink: A system for integrated access to life sciences data sources. IBM System Journal **40(2)** (2001)
5. McEntyre, J.: Linking up with Entrez. Trends Genet. **14(1)** (1998) 39-40
6. Ben Miled, Z., Li, N., Kellett, G., Sipes, B., Bukhres, O.: Complex Life Science Multidatabase Queries. Proceedings of the IEEE **90(11)** (2002)
7. Peim, M., Franconi, E., Paton, N.W., Goble, C.A: Query Processing with Description Logic Ontologies Over Object-Wrapped Databases. Proc. 14th International Conference on Scientific and Statistical Database Management (SSDBM), IEEE Computer Society (2002) 27-36
8. Zdobnov, E. M., Lopez, R., Apweiler, R., Etzold, T.: The EBI SRS server - new features. Bioinformatics **18(8)** (2002) 1149-1150
9. Toner, B.: Rise of the Middle Class: Integration Vendors Differentiate Range of 'N-Tier' Offerings. Bioinform Online, `http://www.bioinform.com` **6(16)**(2002)
10. Wong, L.: Kleisli, a Functional Query System. Journal of Functional Programming **10(1)** (2000) 19-56
11. Paton, N. W., Stevens, R., Baker, P. G., Goble, C. A., Bechhofer, S., Brass, A.: Query Processing in the TAMBIS Bioinformatics Source Integration System. Proc. 11th Int. Conf. on Scientific and Statistical Databases (SSDBM), IEEE Press (1999) 138-147
12. Davidson, S. B., Overton, C., Tanen, V., Wong, L.: BioKleisli: A Digital Library for biomedical Researchers. Journal of Digital Libraries **1(1)** (1997) 36-53
13. Ben Miled, Z., Wang, Y., Li, N., Bukhres, O., Martin, J., Nayar, A., Oppelt, R.: BAO, A Biological and Chemical Ontology For Information Integration. Online Journal Bioinformatics **1** (2002) 60-73

14. Baxevanis, A. D.: The Molecular Biology Database Collection: 2003 update. Nucleic Acids Res. **31(1)** (2003) 1-12
15. Ben Miled, Z., Li, N., Kellett, G., Sipes, B., Bukhres, O.: Complex Life Science Multidatabase Queries. Proceedings of the IEEE **90(11)** (2002)
16. Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., Rapp, B. A., Wheeler, D. L.: GenBank. Nucleic Acids Research **30(1)** (2002) 17-20
17. O'Donovan, C., Martin, M. J., Gattiker, A., Gasteiger, E., Bairoch, A., Pweiler, R.: High-quality protein knowledge resource: SWISS-PROT and TrEMBL Brief. Bioinform. **3** (2002) 275-284
18. Wu, C. H., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Ledley, R. S., Lewis, K. C., Mewes, H., Orcutt, B. C., Suzek, B. E., Tsugita, A., Vinayaka, C. R., Yeh, L. L., Zhang, J., Barker, W. C.: The Protein Information Resource: an integrated public resource of functional annotation of proteins. Nucleic Acids Research **30** (2002) 35-37
19. Falquet, L., Pagni, M., Bucher, P., Hulo, N., Sigrist, C., Hofmann, K., Bairoch, A.: The PROSITE database, its status in 2002. Nucleic Acids Research **30** (2002) 235-238
20. Bairoch A.: The ENZYME database in 2000. Nucleic Acids Research **28** (2000) 304-305
21. Westbrook, J., Feng, Z., Chen L., Yang, H., Berman, H.: The Protein Data Bank and structural genomics. Nucleic Acids Research **31** (2003) 489-491
22. Hamosh A., Scott, A., Amberger, J., Bocchini, C., Valle, D., McKusick, V.: Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. Nucleic Acids Research **30** (2002) 52-55
23. Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breuning, M., Vassalos, V.: Template-Based Wrappers in the TSIMMIS System. In Proceedings of 23rd ACM SIGMOD International Conference on Management of Data (1997) 532-535
24. Crescenzi, V., Mecca, G., Merialdo, P.: ROADRUNNER: Towards Automatic Data Extraction from Large Web Sites. The VLDB Journal (2001) 109-118
25. Knobloc, C., Lerman, K., Minton, S., Muslea, I.: Accurately and Reliably Extracting Data from the Web: A Machine Learning Approach. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering **23(4)** (2000) 33-41
26. Soderland, S.: Learning Information Extraction Rules for Semi-structured and Free Text. Machine Learning **34(1-3)** (1999) 233-272
27. Califf, M. E., Mooney, R. J.: Relational Learning of Pattern-Match Rules for Information Extraction. Proceedings of AAAI Spring Symposium **6-11** (1996)
28. Cohen, W.: Text categorization and relational learning. Proceedings of the 12th International Conference on Machine Learning (1995) 124-132
29. Kushmerick, N., Weld, D. S., Doorenbos, R.: Wrapper Induction for Information Extraction," International Joint Conference on Artificial Intelligence (IJCAI) (1997) 729-737
30. Hammer, J., Garcia-Molina, H., Cho, J., Aranha, R., Crespo, A.: Extracting Semistructured Information from the Web. Proceedings of the 1st Workshop on Management for Semistructured Data (1997) 18-25
31. Huck, G., Frankhausewr, P., Aberer, K., Neuhold, E.: Jedi: Extracting and Synthesizing Information from the Web. Proceedings of Conference on Cooperative Information Systems (1998) 32-43

Efficient Techniques to Explore and Rank Paths in Life Science Data Sources*

Zoé Lacroix[1], Louiqa Raschid[2], and Maria-Esther Vidal[3]

[1] Arizona State University, Tempe, AZ 85287-6106, USA,
zoe.lacroix@asu.edu
[2] University of Maryland, College Park, MD 20742, USA,
louiqa@umiacs.umiacs.umd.edu
[3] Universidad Simón Bolívar, Caracas 1080, Venezuela,
mvidal@ldc.usb.ve

Abstract. Life science data sources represent a complex link-driven federation of publicly available Web accessible sources. A fundamental need for scientists today is the ability to completely explore all relationships between scientific classes, e.g., genes and citations, that may be retrieved from various data sources. A challenge to such exploration is that each path between data sources potentially has different domain specific semantics and yields different benefit to the scientist. Thus, it is important to efficiently explore paths so as to generate paths with the highest benefits. In this paper, we explore the search space of paths that satisfy queries expressed as regular expressions. We propose an algorithm ESearch that runs in polynomial time in the size of the graph when the graph is acyclic. We present expressions to determine the benefit of a path based on metadata (statistics). We develop a heuristic search OnlyBestXX%. Finally, we compare OnlyBestXX% and ESearch.

1 Introduction

A fundamental problem facing the scientist today is correctly identifying a specific biological instance, e.g., a specific gene or protein, and then obtaining a complete functional characterization of this object. Such characterizations are critical in various scientific tasks including collecting data for an experiment, validation of hypotheses, and data curation. A complete characterization of a scientific entity class typically involves the exploration of several distributed resources often linked in a federation. This exploration may be supported by a query involving multiple joins, combining data from various resources, followed by filtering and ranking. There are many challenges in solving this problem when exploring an inter-related and inter-linked federation of sources.

While each source may have data on one or more scientific entity classes such as genes and sequences, there is significant diversity in the *coverage* of these

* This research has been partially supported by the NIH National Institute on Aging under grant number 1 R03 AG21671-01 and the National Science Foundation under grants IIS0219909 and EIA0130422.

sources. For example, the NCBI and EMBL Nucleotide databases have different attributes characterizing (describing) sequences although they both cover the same sequences. In contrast, AllGenes, RatMap and the Mouse Genome Database (MGD) are all gene databases but they have different coverage. AllGenes contains human and mouse genes and overlaps with MGD. See [12] for a detailed discussion on various aspects of coverage.

A second challenge is that for various scientific tasks, scientists are interested in exploring all relationships between scientific entity classes, e.g., genes and citations. These relationships are often implemented as physical links between objects in the data sources. Each link between sources may be visualized as a collection of individual links, going from a data object in one source to another data object, in the same or a different source. The properties of the link may vary, e.g., uni- or bi-directional, cardinality such as 1:1 or 1:N, etc.

An exploration process typically starts from one or more available sources, and continues by following direct links, e.g., URLs, or traversing paths, informally, concatenations of links via intermediate sources. Thus, given some start class in source S and target class in source T, there may be multiple alternate paths to navigate from S to T.

Given a set of paths from a start entity class to a target class, one could thus differentiate between these paths and rank these paths relevant to the benefit associated with some domain specific criteria. Example benefit criteria include maximizing result cardinality of target objects; maximizing the number of attributes along the path, maximizing the number of sources that were visited in the path, etc. Such benefits can be exploited by the scientist in choosing paths of interest. They can also be exploited by a query optimizer. For example, one can predict the cost of evaluating a query given some specific sources and paths.

In this paper, we present an approach to efficiently explore life sciences data sources. While typical scientific discovery usually involves complex queries involving multiple joins, selections and projections, and ranking of results, in this paper, we focus on one important aspect: the navigational aspect of exploring links and paths. Thus, we initially consider a regular expression based query language that captures this navigational aspect. In this simplified context, a solution to a query (regular expression) is a path in a graph of sources and links between sources that satisfies the regular expression. The problem of determining if a pair of nodes in a graph occurs in a path that satisfies a regular expression has been shown to be NP complete [9]. We propose an algorithm based on a deterministic finite state automaton (DFA) that recognizes a regular expression. The algorithm *ESearch* performs an exhaustive breadth-first search of all paths in a graph. The algorithm runs in polynomial time in the size of the graph when the graph is acyclic.

We next consider various criteria to determine the benefit of a path. Examples are the path visiting the largest number of sources, or the maximum number of attributes, or the path with maximum result cardinality. Note that these may correspond to well known NP problems, e.g., longest path. We provide expressions to estimate the benefit of a path based on the metadata (statistics) of a source graph *SG* and the Object Graph (data objects and links between objects). We rank the paths produced by *ESearch* based on the metadata benefit and identify those paths in the top 25% as the set of *Relevant Paths.*

ESearch is an exhaustive search and explores all matching sources. To be more efficient, we develop a heuristic search, *OnlyBestXX% Search,* that resembles a best-first search strategy. Informally, the heuristic search will rank all sources in the *SG* based on a utility measure. Potential candidate utility measures include the *image cardinality* of a source (number of objects participating in an inlink) and the *link cardinality* (number of outgoing links). Intuitively, we expect that a source with a high utility measure will participate in a path with a high benefit. We note that there could be many relevant utility measures, depending on the specific benefit. We compare the *precision, recall* and *fallout* of *OnlyBestXX% Search* with respect to the Relevant Paths of *ESearch.* While *OnlyBestXX% Search* can be more efficient, we demonstrate the negative impact when the utility measure mispredicts the choice of good sources. This occurs both for *OnlyBest25%* and *OnlyBest50%* which choose fewer sources, as well as for *OnlyBest75%* and *OnlyBest90%* which choose more sources. This motivates the need for a *Best Subpath First* search that continually ranks and explores the best subpaths with the highest benefit, rather than relying on a heuristic that only chooses good sources.

Our research on semantics complements work in [8] that studies the properties associated with multiple alternate paths, e.g., result cardinality, as well as overlap of results in multiple alternate paths. These two projects provide a strong foundation for efficiently exploring paths.

There has been prior research on providing access to life science sources [2, 3, 7, 16]. Example of such systems include DiscoveryLink [6], Kleisli and its successors [1, 17], SRS [4, 5] and TAMBIS [13]. Typically, these systems have not explored alternate sources and multiple paths. Recent research in [10, 11] has addressed the problem of alternate sources for scientific entity classes. They consider a more expressive query language PQL, in comparison to the language limited to regular expressions defined in this paper. However, they do not consider heuristics to explore the search space or the semantics of multiple alternate paths. Since we are interested in the navigational aspects of queries, we focus on the regular expression based language. Properties of links have been studied in the context of XML document processing [14, 15]; they do not consider semantics associated with paths. Our contributions are as follows:

- We present a regular expression based query language to explore the navigational aspects of scientific queries against the graph of life science sources. We develop an exhaustive search algorithm *ESearch* to find paths that satisfy regular expressions.
- We present domain specific semantic criteria to compare and rank alternate paths. We rank the paths produced by *ESearch* based on metadata benefits and identify the top 25% as *Relevant Paths.*
- We develop a heuristic search *OnlyBestXX%* which uses a metadata based utility measure to only explore some sources. We compare the precision,

recall and fallout of *OnlyBestXX%* with respect to the *Relevant Paths* (with the highest benefit) produced by *ESearch.*

2 Physical and Logical Map of Life Sciences Data Sources

Life science sources may be modeled at two levels: the physical and logical level. The physical level corresponds to the actual data sources and the links that exist between them. An example of data sources and links is shown in Figure 1. The physical level is modeled by a directed graph *SG,* where nodes represent data sources and edges represent a physical implementation of a link between two data sources. A data object in one data source may have a link to one or more data objects in another data source, e.g., a gene in GeneCards links to a citation in PubMed. A path in *SG* is defined in a straightforward manner by traversing the links of *SG.*

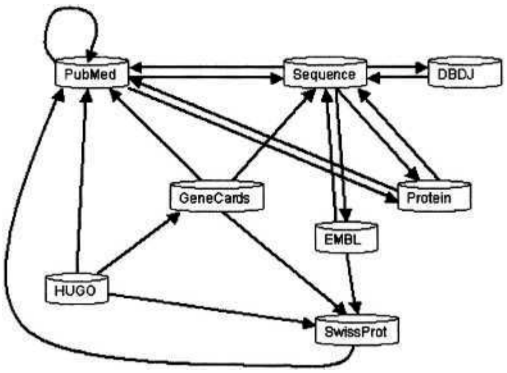


Fig. 1. A graph *SG* of Life Science Sources

The logical level consists of classes (entity classes, concepts or ontology classes) that are implemented by one or more physical data sources or possibly parts of data sources. For example, the class *Citation* may be implemented by the data source PubMed. Each source typically provides a unique identifier for the entities of a class and includes attribute values that characterize them. An example of a possible (commonly accepted) mapping from logical classes to data sources is illustrated in Table 1.

3 Simple Regular Expression Based Query Language

First, we present a complex query typical of scientific discovery and describe how it may be evaluated on life sciences sources. While the efficient evaluation of such complex queries is important, in this paper, we focus on the navigational

CLASS	DATA SOURCE
Sequence (s)	NCBI Nucleotide database EMBL Nucleotide Sequence database DDBJ
Protein (p)	NCBI Protein database Swiss-Prot
Citation (c)	NCBI PubMed

Table 1. A Possible Mapping from Logical Classes to Physical Data Sources

aspect of exploring multiple alternate links and paths. Thus, we consider a simple language based on regular expressions. While we recognize that such a language has limited expressive power, it is sufficient for us to express domain specific semantics that differentiate among alternate paths in *SG.* Consider the following complex query:

> **Query 1:** *Return all citations of PubMed published since 1995 that mention "heart" and refer to sequences of GenBank that are annotated as "Calcium channel".*

The task of scientific discovery encapsulated in this query requires knowledge of the capabilities of sources. For example, PubMed and GenBank accept queries containing multiple selection predicates. They perform database-like selection (by retrieving only the entries relevant to the filter and all associated information) on the unique identifiers of entries in PubMed or GenBank. They also support a more sophisticated keyword based search capability. Constraints such as `date since 1995` can also be used to filter the results. This increases the query capability and can improve the efficiency of access. Both the keyword `heart` and the constraint `date since 1995` can be passed to PubMed to filter the results. PubMed typically removes duplicates when they occur in the output. Sending a single request with say multiple keywords to a remote source may return a result containing duplicates which must be eliminated locally. However, this alternative (while including duplicate results) may be less expensive than submitting multiple calls on the Web, one request for each keyword.

To answer this query, one can access PubMed and retrieve citations published since 1995 that mention "heart", and then extract all GenBank identifiers that they contain. Next, one can retrieve the information available in GenBank for each sequence and filter the ones that are annotated as "calcium channel". Alternately, one can follow the *Nucleotide* link of Entrez PubMed.

A different plan would first access GenBank and retrieve all sequences that are annotated as "Calcium channel", and then extract the MEDLINE identifiers for these. Next, one can retrieve the corresponding MEDLINE citations from PubMed and filter the ones published since 1995 that mention "heart". A third alternative consists in querying PubMed and GenBank concurrently and performing an appropriate join of the retrieved data.

This example illustrated many aspects of identifying sources, selecting source query processing capabilities, selecting the order of accessing sources, choosing to follow a link versus evaluating a join, etc. While all these aspects are important, we focus on the navigational aspects of following multiple alternate links and paths. To do so, we consider a regular expression based query language.

We consider regular expressions expressed (RE) over class names in set *E*. Given an input regular expression r the objective is to interpret r on the graph *SG*. Class labels in *E* include p (protein), s (sequence), g (gene), c (citation), and ϵ (a wild card label). Each class label may have multiple interpretations in *SG*. We consider the data sources NCBI Nucleotide, EMBL Nucleotide Sequence, DDBJ, NCBI Protein, Swiss-Prot, HUGO, GeneCards, and PubMed as shown in Figure 1. Table 2 describes the interpretation of regular expressions by paths in *SG*. Paths of length 0 are expressed by a single class name of *E* (or a disjunction of class names) and interpreted by sources of the graph *SG* by a mapping from *E* to *SG* as illustrated in Table 1. Paths of length 1 are expressed by the concatenation of two class names and are interpreted by a physical link between two sources of *SG*.

RE	Interpretation in SG
e_i, for all e_i in E	A path comprising a node in SG that implements e_i
ϵ	A path comprising any node in SG
$e_i \,.\, e_j$	A path comprising any edge in SG between 2 nodes that implement e_i, e_j.
$r_i.r_j$	Path $p_i.p_j$ where p_i (p_j) interprets r_i (r_j).
$r_i \vert r_j$	Path p_i or p_j where p_i (p_j) interprets r_i (r_j).
r^+	$p_1.p_2 \cdots p_n$ where $n \geq 1$ and each p_i is a path (node) that interprets r

Table 2. Mapping from Regular Expressions to *SG*

Regular expressions express common retrieval queries. For example, a scientist may be interested to *"Retrieve citations linked to genes via any number of intermediate sources (paths of any length* $\geq$ *2)"*.

Query 2: $g.\epsilon^+.c$

This query will match any path that starts with a node that interprets g and terminates in a node that interprets *c*. The length of that path can vary from 2 to the length of the longest path in *SG* (if this path satisfies the expression). Thus, we see that enumerating all the paths in *SG* that match the query could be exponential in the size of *SG*.

A query such as **Query 2** is important to scientists who wish to fully characterize scientific objects without specifying the sources and paths to visit. Evaluating such a query is challenging since in order to obtain the most complete list of citations relevant to a gene, one would need to visit all relevant paths. However, for efficiency, one would wish to avoid all irrelevant paths. Section 4 presents algorithms to develop solutions and Section 5 presents semantic criteria to rank solutions.

4 Complexity of Exploring Paths in *SG*

4.1 Paths Satisfying a Regular Expression

Definition 1.
A graph $SG = (S, L)$ is a directed acyclic graph, where:

- S is a set of nodes where each node corresponds to exactly a source;
- L is a set of edges $L \subset S \times S$ that represents links between sources.

The graph is mapped to the logical level as follows:

- ϕ is a mapping from a class name in E to a set of physical sources in S. ϕ is a one-to-many mapping as illustrated in Table 1. The wild card ϵ is mapped to S.
- γ is a mapping from a source in 5 to a class name in E. Note that each source of S is mapped by γ to a single class name in E, or to the wild card ϵ.

Note that each node (source) implements one entity class; there is at most one edge between any two nodes; there are no cycles in the graph.

A path $p = (s_1, s_2, \ldots, s_n)$ in SG is defined as a list of sources $s_i \in S$ and is mapped to a regular expression $\gamma(p)$ defined by $\gamma(s_1).\gamma(s_2)\ldots\gamma(s_n)$, i.e., the concatenation of $\gamma(s_i) \in E$. A regular expression r over the alphabet E expresses a retrieval query Q_r as shown in Section 3. The result of Q_r is the set of paths p in SG that interpret r, that is $\{p \in SG \mid \phi(r) = p\}$.

4.2 Searching the Space of Paths

A naive method for evaluating a query Q_r on SG is to traverse *all* paths in SG, and to determine if they interpret r. The time complexity of the naive evaluation is exponential in the size of SG because SG has an exponential number of paths. A similar problem was addressed in [9] where it was shown that for (any) graph and regular expression, determining if a particular edge occurred in a path that satisfied the regular expression and was in the answer was NP complete.

The ESearch Algorithm
ESearch is based on a deterministic finite state automaton (DFA) that recognizes a regular expression r. The algorithm performs an exhaustive breadth-first search of all paths in a graph. Suppose DFA is the automaton that recognizes the regular expression r. The DFA is represented by a set of transitions, where a transition is a triple $t=(i,f,e)$, where, i represents the initial state of t, f represents the final state of t and, e corresponds to the label of t ($e \in E$, the set of class names). The state i (resp. f) may be a *start state* (resp. *end state*) of DFA.

The exhaustive algorithm *ESearch* comprises two phases: (a) *build path* and (b) *print path.* In phase *build path,* for each visited transition t=(i,f,e), the algorithm identifies *all* the sources $s_i \in \phi(e)$. If i is not a *start state* of the DFA, then, for each s_i, the algorithm computes a set $s_i.previousSources$. To do so, it considers all the sources that were selected in transition t^p previous to t, and selects the subset of sources that are adjacent to s_i in *SG;* these sources are included in $s_i.previousSources$. In phase *print path,* the algorithm starts from the set of sources corresponding to the final transition, whose final state is an *end state* of the DFA. For each s_i, it uses the set $s_i.previousSources$ to construct a path. The path terminates in one of the sources visited by the start transition whose initial state is a *start state* of the DFA. We note that *print path* may commence as soon as *ESearch* visits the first transition for which f is a final state of DFA. In our implementation, we do not consider such potential parallelism between these two phases.

The *ESearch* Algorithm runs in polynomial time in the size of the graph, if graph is cycle free and all paths are cycle free. Each node (source) in the graph implements only one class, so a node is visited at most once in each transition (each level of the breadth-first search). Similarly, each node is visited at most once in each iteration of *print path.* An annotated *SG* is produced during the *build path* phase of *ESearch.* For each transition in the DFA, each source s_i matching the transition is annotated with sources in $s_i.previousSources$. If d is the maximum number of sources that can precede a source in the annotated *SG,* i.e., the cardinality of *previousSources,* and b is the maximum length of (cycle free) paths satisfying the regular expression, then $O(b^d)$ is an upper bound for *ESearch.*

Finding paths that satisfy a regular expression has some similarity to join ordering in relational queries and capability based rewriting. The complexity of the join ordering problem depends on the size of the query (e.g., number of joins), whereas the complexity of finding paths depends on the graph topology. Further, join ordering is a syntactic rewriting task since the join order does not impact the results. In our case, each path may produce different results. Our problem is similar to the capability based rewriting for queries, in a *global as view* scenario. A common limited capability is bindings between conjunctive subqueries which must be respected during rewriting. The presence of the wild card ϵ and the (*) operator in regular expressions makes our problem more complex. We may need to consider paths of any length in the graph. This problem is similar to answering queries using views.

5 Ranking Solutions Using Semantic Criteria

Consider an example where two sources characterize an object using different attributes. Consider the query *"Retrieve all information known about gene TP53".* It can be expressed by the regular expression g which can be interpreted by two sources GeneCards and GeneLynx. Using the fulltext search engine on GeneCards with the keyword `TP53` returns 24 hits, whereas a similar query to

GeneLynx only retrieves 3 hits. The information (number of attributes and their values) contained in the 24 GeneCards entries about this gene TP53 is richer than the information in the 3 entries retrieved from GeneLynx.

Similar to the example above, each path through the Source Graph may have different properties and will be of different benefit. Next, we enumerate *domain specific criteria* that can be used to express the benefits of paths and that can be used to rank all paths that satisfy some regular expression.

5.1 Criteria to Rank Paths

We focus on metadata benefits that are based on statistics from the Source Graph and the Object Graph (data objects and links between objects). One could also develop benefits that reflect other properties from the Source Graph (schema), e.g., paths that include some particular source such as GenBank, or paths that reflect particular objects and links, e.g., paths that include the gene TP53. We do not consider such benefits in this paper. The following are some potential criteria for ranking paths:

– Path length semantics: [Minimize/Maximize] the length of the path. Since a source may implement multiple classes, a source could be visited multiple times when traversing a path. Thus, an alternate criteria is [Minimize/Maximize] the number of distinct sources that are visited.
– Attribute cardinality semantics: [Minimize/Maximize] the (total) number of attributes of all classes along the path. An alternative is [Minimize/Maximize] the number of attributes for a specific class in the path.
– Result cardinality semantics: [Minimize/Maximize] the cardinality of the results (the number of retrieved entries). This could refer to the cardinality of any of the intermediate or final classes. In the next section, we develop expressions for Path Cardinality and Target Object Cardinality.

Related research in [10, 11] did consider alternate paths. However, they focused on the case where the *ideal* relationship between objects was 1 to 1. We consider more general cases where the relationship could be 1 to N or N to M. They also did not consider ranking the paths.

5.2 Metadata Benefit of a Path

We now provide expressions to estimate Target Object Cardinality and Path Cardinality. Consider a path p through sources $S_1, S_2, \ldots, S_n$.

– Source cardinality: $c(S_i)$ is the number of data objects in source S_i.
– Link cardinality: $l(S_{i,i+1})$ is the number of links from all data objects of source S_i pointing to data objects of S_{i+1}.
– Link participation (start source): $l_{par}(S_{i,i+1})$ is the number of objects in S_i having at least one outgoing link to an object in S_{i+1}.
– Link image (target source): $l_{im}(S_{i,i+1})$ is the number of data objects in S_{i+1} that have at least one incoming link from objects in S_i.

Estimating the benefit of a path given metadata from the Source Graph is not straightforward. Naumann et al developed a model [8] for such estimates. It made the following assumptions, for any subpath i composed of edges (S_{i-1}, S_i) and (S_i, S_{i+1}): (1) Uniform distribution of the links among the objects. (2) Object independence for objects participating in a link, i.e., the probability for an object in S_i to have an outlink is independent of all other objects in the source. (3) Path independence for each subpath i, S_{i-1}, S_i, S_{i+1}, i.e., the probability for an object in S_i to have an inlink from an object in S_{i-1} is independent of the probability of the object having an outlink to an object in S_{i+1}.

Path Cardinality

The metadata benefit *Path Cardinality* for path p, $PC(p)$, where $n >= 3$, can be expressed as follows:

$$PC(p) = l(S_{1,2}) \times \prod_{i=2,\ldots,n-1} [pcf(i)]. \tag{1}$$

If we assume uniform distribution, object independence and path independence, then we substitute the following for the path cardinality fraction: $pcf(i) = l(S_{i,i+1})/c(S_i)$. If we do not assume path independence and assume that each incoming link is to an object with an outgoing link, then we have the following: $pcf(i) = l(S_{i,i+1})/l_{par}(S_i)$.

Target Object Cardinality

The metadata benefit *Target Object Cardinality* for path p, can be expressed as follows:

$$TOC(p) = c(S_n)tocf(n) \tag{2}$$

The $tocf(i)$ is a factor representing the probability that an object in S_i can be reached from some object in S_1.

The value of $tocf(i)$ for a path from S_1 to S_2 is trivially $tocf(2) = l_{im}(S_{1,2})/c(S_2)$.

We now derive an expression for $tocf(i+1)$ given some value for $tocf(i)$. An object x in S_{i+1} receives on average $\delta^{in}_{i+1} = l(S_{i,i+1})/l_{im}(S_{i,i+1})$ edges from objects in S_i. If at least one of these objects in S_i is reached, then we will reach x. Similarly, an object x will not be reached if all δ^{in}_{i+1} objects in S_i are not reached. To compute $tocf(i+1)$ we compute the probability that an object is not reached for some $tocf(i)$, and then use that to compute the probability that it is reached. Thus, we have the following:

$$tocf(i+1) = (l_{im}(S_{i,i+1})/c(S_{i+1})(1 - (1 - tocf(i))^{\delta^{in}_{i+1}}) \tag{3}$$

5.3 Benefits of Paths Produced by ESearch

Our experiments for *ESearch* and the heuristic searches were run on a graph *SG*(Definition 1) with 26 entities. The number of nodes and edges were varied from a sparse *SG* with 111 nodes and 272 edges to a dense *SG* of 125 nodes and 1845 edges. The image cardinality and the link cardinality for the edges were drawn from some normal distributions. In order to simulate a mixed workload, we created 20 regular expression queries of varying expressive power, and we report on the total number of solutions (paths) aggregated over all the queries. We also report on the top 25% of the solutions or *Relevant Paths*.

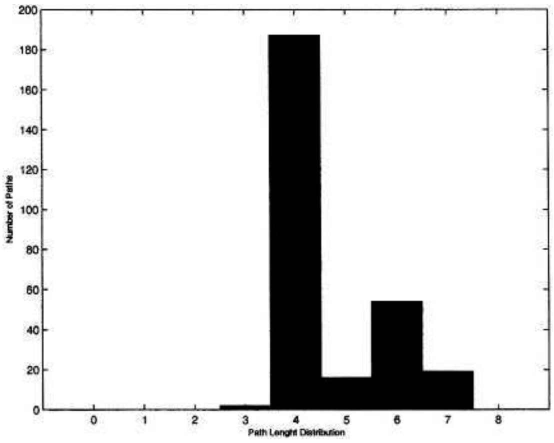


Fig. 2. Length of Paths for One Query

We report here on the metadata benefits of the paths for a randomly selected query among the 20 queries for the dense *SG*. *ESearch* produced 278 paths for this query. A histogram of the path length is reported in Figure 2. As can be seen, path lengths varied from 3 to 7 with 4 being the most frequent. The Path Cardinality for all the paths produced by *ESearch* ranged from 3.4E+5 to 4.3E+12 while the Path Cardinality for *Relevant Paths* (top 25%) ranged from 1.5E+10 to 4.3 E+12. The Target Object Cardinality for all the paths ranged from 1.0 to 5.0E+8 while the Target Object Cardinality for *Relevant Paths* ranged from 4.7E+5 to 5.0E+8.

6 Heuristics to Efficiently Explore the Search Space

6.1 OnlyBestXX% Search

As will be seen from our experiments, *ESearch* has to explore all subpaths of length 1, 2, etc., up to the longest path, before it can start producing any paths. In order to make the search more efficient, we develop a heuristic search algorithm *OnlyBestXX% Search;* it is based on a utility measure that ranks the sources in *SG* and only selects the best XX% sources. We chose a utility measure to correspond to the metadata benefit. When the metadata benefit is Target Object Cardinality, the *OnlyBestXX% Search* uses a utility measure based on the image cardinality of each source. All sources S_j in *SG* are ranked based on the image cardinality of inlinks (S_i, S_j). When the metadata benefit is Path Cardinality, the ranking of sources S_i is based on the link cardinality of the outlinks (S_i, S_j).

We surmise that when exploring paths, selecting a source that ranks high on a utility measure using image or link cardinality, respectively, will lead to paths with higher benefits for Target Object Cardinality and Path Cardinality, respectively. This choice of utility measure is reasonable since for example the Target Object Cardinality indeed depends on the image cardinality. We note that the benefit actually depends on the image cardinality of *all* the sources in the path and the utility measure is making a local rather than a global choice.

The algorithm *OnlyBestXX%* modifies *build path* of *ESearch.* During the phase *buildpath,* all sources satisfying the current transition t are ranked based on either the image or the link cardinality. Instead of considering *all* sources $s_i \in \phi(e)$, only those s_i that occur in the *top XX%* of the ranking are chosen. For each winner $s_{i,n}$, where $1 \leq n \leq K$, the set $s_{i,n}.previousSources$ is only initialized from among the set of *top XX%* winners visited at the previous transition, that are adjacent to $s_{i,n}$ in *SG.*

An ideal search would perform a look ahead to determine the utility measure of all potential sources that may occur in the path. In contrast, our heuristic search is not perfect and will only select the best sources that match a particular transaction. Since the heuristic does not have a look ahead and cannot consider all sources in the path, it may also make poor choices in an early stage and pay the penalty for this choice. Further, it is not exhaustive, and can fail to produce a path depending on the choice of *top XX%* winners $s_{i,n}$ and $s_{i,n}.previousSources$.

6.2 Comparison of ESearch and OnlyBestXX%

We report on a comparison of *ESearch* and *OnlyBestXX%* on a graph *SG* that varies from sparse to dense as described earlier. We report on results averaged over 20 randomly generated regular expression queries. We first compare the efficiency of *ESearch* and *OnlyBestXX%* to produce paths. We then compare the precision, recall and fallout of *OnlyBestXX%* with respect to *Relevant Paths,* the top 25% of paths produced by *ESearch.*

Figure 3 reports on the efficiency of the algorithms in producing paths. The X-axis reports on the number of iterations, i.e., the number of nodes that were visited by the algorithm in the *buildpath* phase[4]. The Y-axis reports on the total number of successful solutions (paths)[5] that were generated by the search[6].

[4] Note that a node can be visited multiple times by the algorithm, hence the number of visits (iterations) can exceed the absolute number of nodes in *SG.*

[5] Each point in the graphs is obtained by averaging over 20 queries generated randomly.

[6] Recall that we did not consider potential parallelism between the two phases *build path* and *print path.* Hence, we first allow *build path* to iterate over a given number of visits, say 500, and then report on the total number of solutions found by visiting these 500 nodes.

Figure 3(a) corresponds to a sparse graph. *OnlyBest25%* is the first to produce paths, followed by *OnlyBest50%* and *OnlyBest75%,* etc. *ESearch* does not begin to produce paths until visiting 100+ nodes. Since *OnlyBest25%* and *OnlyBest50%* consider fewer sources, they eventually are not able to produce new paths. Eventually *ESearch* produces more paths than all the other algorithms. Figure 3(b) corresponds to a dense graph. Here it is clear that while *ESearch* is slow to produce paths, it quickly outperforms all the other algorithms.

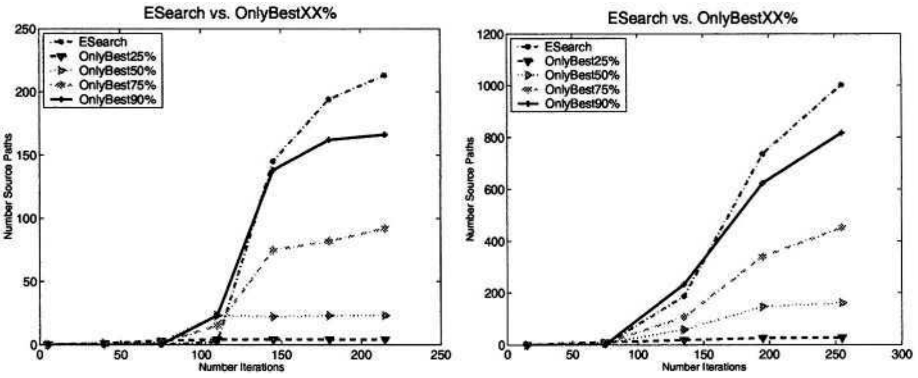


Fig. 3. Comparison of ESearch and OnlyBestXX% for 20 regular expressions for a sparse (a) and dense (b) *SG*

6.3 Precision, Recall, and Fallout of OnlyBestXX% Search

Relevant paths:
We order the paths produced by *ESearch* based on Target Object Cardinality or Path Cardinality, respectively. The top 25% are the *Relevant Paths.* We report precision, recall and fallout of the *OnlyBestXX%* algorithms with respect to these *Relevant Paths.* These results are for a dense *SG.*
Precision:
This is the fraction of paths produced by the *OnlyBestXX%* algorithm that are *Relevant Paths.* Precision reflects how much of the work done by the search was useful in producing *Relevant Paths.* Since each *OnlyBestXX%* algorithm produces a different number of paths, we also plot the absolute number of *Relevant Paths* for comparison. Figure 4(a) reports on the precision and Figure 4(b) reports on the number of paths. The number of paths produced increases, as the heuristic chooses more sources, e.g., *OnlyBest90%* produces the most number of relevant paths. However, we see that the precision for the different algorithms ranges from 0.2 to 0.35. In some cases, the precision increases from *OnlyBest25%* to *OnlyBest90%* but this behavior is not consistent. To explain, *OnlyBest25%* and *OnlyBest50%* choose fewer sources and produce fewer paths. The utility measure is not perfect at choosing the best sources. Consequently, the impact

of mispredicting the choice of a good source will have a negative impact on *OnlyBest25%* and *OnlyBest50%*. In contrast, *OnlyBest75%* and *OnlyBest90%* choose more sources and produce more paths. The negative impact of mispredicting a good source is that many paths that are not in *Relevant Paths* may be produced.

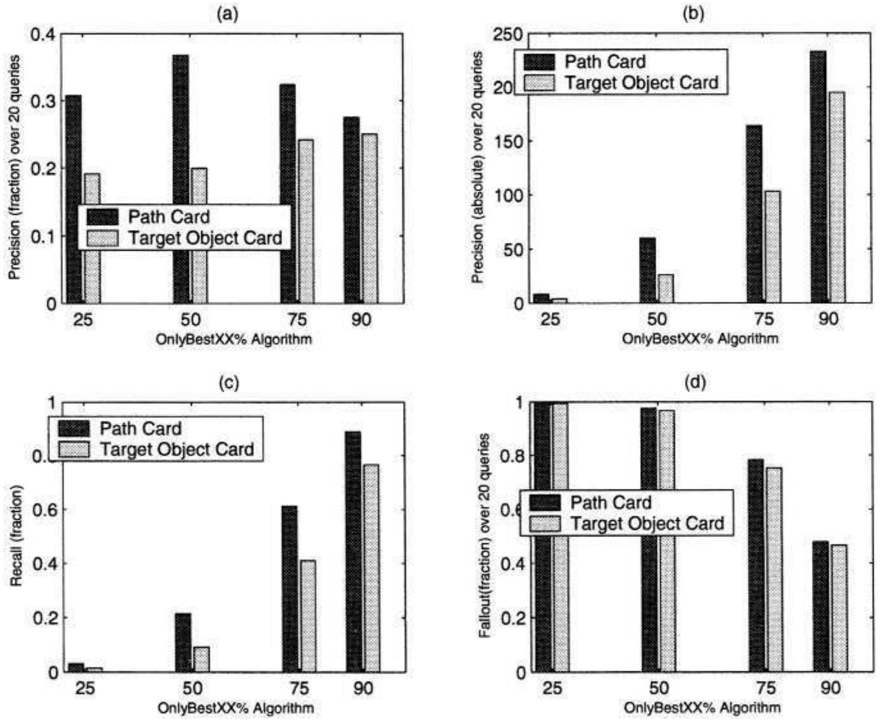


Fig. 4. Precision, Recall and Fallout for *OnlyBestXX% Search*

Recall: This is the fraction of paths that are *Relevant Paths* and are produced by an *OnlyBestXX%* algorithm. This fraction reflects the coverage of the *OnlyBestXX%* algorithm with respect to *ESearch* which produces all *Relevant Paths* and has a recall of 1.0. Figure 4(c) reports on the recall. As expected *OnlyBest75%* and *OnlyBest90%* have better recall since they produce more paths. We note that while *OnlyBest75%* chooses up to 75 % of the matching sources, its recall of *Relevant Paths* ranges between 0.4 and 0.6 for the two benefits. This implies that while the utility measure's choice of the best sources is reasonable, since the heuristic does not consider all the sources in the path, there is still some negative impact of misprediction.
Fallout:
The fraction of those paths (75% of *ESearch*) that are not *Relevant Paths* and were not produced by the *OnlyBestXX%* Algorithm. Fallout corresponds to the

efficiency in avoiding paths not in *Relevant Paths* and reflects the impact of misprediction of the best sources by the utility measure; it is reported in Figure 4(d). A perfect search (*ESearch*) has a fallout of 1.0. We note that OnlyBest25% and OnlyBest50% have a high fallout. This implies that when the number of sources chosen are small, the algorithm did reasonably well in avoiding paths not in *Relevant Paths*. For *OnlyBest75%* and *OnlyBest90%* the fallout increases. It is 0.5 for *OnlyBest90%* and indicates lost effort. This reflects that in the case where more sources are chosen and more paths are produced, the misprediction of the utility measure in choosing the best sources can have significant impact.

To summarize, while *OnlyBestXX%* can produce paths more quickly than *ESearch,* there may be potential negative impact when the utility measure mispredicts the choice of good sources. This occurs both for *Only Best25%* and *Only Best50%* which choose fewer sources, as well as for *Only Best75%* and *Only Best90%* which choose more sources. In the former case, less paths in *Relevant Paths* are produced and in the latter case, more paths that are not in *Relevant Paths* may be produced. This motivates that heuristics be developed that can choose different subpaths to explore at various steps of the search.

7 Summary and Conclusions

We use queries based on regular expressions to characterize the navigational aspects of scientific exploration. We describe an exhaustive breadth-first search *ESearch* based on a determinist finite state automaton (DFA); it runs in polynomial time in the size of the graph when the graph is acyclic. We present expressions to determine the metadata benefit of a path and rank the paths produced by *ESearch*; the top 25% is the set of *Relevant Paths.*

We then present a heuristic search, *OnlyBestXX% Search.* It selects a subset of sources using a utility measure based on image cardinality or link cardinality, respectively. We compare the *precision, recall* and *fallout* of *Only BestXX% Search* with respect to the *Relevant Paths* of *ESearch.* We show that while the utility measure is reasonable in choosing good sources, there is a negative impact of misprediction, both for *OnlyBest25%* and *OnlyBest50%* that choose few sources and for *OnlyBest75%* and *OnlyBest90%* that choose more sources. This motivates the need for a *Best Subpath First* search that continually ranks and explores the best subpaths with the highest benefit, rather than relying on a heuristic that only chooses good sources.

In future work, we will test our model to estimate metadata benefits using statistics from real data sources. We will also define benefits based on criteria that reflect the schema of the Source Graph and the identity of objects and links in the Object Graph.

Acknowledgments We thank George Mihaila, Felix Naumann and Julian Mestre for many fruitful discussions and Julian Mestre and WeiWei Zhang for programming suport.

References

[1] S. Davidson, J. Cabtree, B. Brunk, J.Schug, V. Tannen, C. Overton, and C. Stoeckert. K2/kleisli and gus: Experiments in integrated access to genomic data sources. *IBM Systems Journal,* 40(2), 2001.

[2] B. Eckman, A. Kosky, and L. Laroco. Extending traditional query-based integration approaches for functional characterization of post-genomic data. *BioInformatics,* 17(2), 2000.

[3] B. Eckman, Z. Lacroix, and L. Raschid. Optimized seamless integration of biomolecular data. *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering,* 2001.

[4] T. Etzold and P. Argos. Srs: An indexing and retrieval tool for flat file data libraries. *Computer Applications of Biosciences,* 9(1), 1993.

[5] T. Etzold and G. Verde. Using views for retrieving data from extremely heterogeneous databanks. *Pacific Symposium on Biocomputing,* pages 134–141, 1997.

[6] L. Haas, P. Kodali, J. Rice, P. Schwarz, and W. Swope. Integrating life sciences data - with a little garlic. *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering,* 2000.

[7] G. Kemp, C. Robertson, and P. Gray. Efficient access to biological databases using corba. *CCP11 Newsletter,* 3.1(7), 1999.

[8] Z. Lacroix, H. Murthy, F. Naumann, and L. Raschid. Links and paths through life sciences data sources. *In* Proceedings of the International Workshop on Data Integration for the Life Sciences (DILS), *Leipzig, Germany,* 2004.

[9] Alberto O. Mendelzon and Peter T. Wood. Finding regular simple paths in graph databases. In *Proceedings of the International Conference on Very Large Data Bases,* pages 185–193, 1989.

[10] P. Mork, A. Halevy, and P. Tarczy-Hornoch. A model for data integration systems of biomedical data applied to online genetic databases. *Proceedings of the AMIA,* 2001.

[11] P. Mork, R. Shaker, A. Halevy, and P. Tarczy-Hornoch. Pql: A declarative query language over dynamic biological data. *Proceedings of the AMIA,* 2002.

[12] Felix Naumann. *Quality-driven Query Answering for Integrated Information Systems,* volume 2261 of *Lecture Notes on Computer Science (LNCS).* Springer Verlag, Heidelberg, 2002.

[13] N.W. Paton, R. Stevens, P.G. Baker, C.A. Goble, S. Bechhofer, and Brass. Query processing in the tambis bioinformatics source integration system. *Proceedings of the IEEE Intl. Conf. on Scientific and Statistical Databases (SSDBM),* 1999.

[14] N. Polyzotis and M. Garofalakis. Statistical synopses for graph-structured xml databases. *Proceedings of the ACM SIGMOD Conference,* 2002.

[15] N. Polyzotis and M. Garofalakis. Structure and value synopses for xml data graphs. *Proceedings of the Very Large Data Base Conference,* 2002.

[16] T. Topaloglou, A. Kosky, and V. Markovitz. Seamless integration of biological applications within a database framework. *Proceedings of the Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB),* 1999.

[17] L. Wong. Kleisli: Its exchange format, supporting tools, and an application protein interaction extraction. *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering,* 2000.

Links and Paths through Life Sciences Data Sources

Zoé Lacroix[1], Hyma Murthy[2], Felix Naumann[3], and Louiqa Raschid[2]

[1] Arizona State University
zoe.lacroix@asu.edu
[2] University of Maryland
louiqa,hmurthy@umiacs.umd.edu
[3] Humboldt-Universität zu Berlin
naumann@informatik.hu-berlin.de

Abstract. An abundance of biological data sources contain data on classes of scientific entities, such as genes and sequences. Logical relationships between scientific objects are implemented as URLs and foreign IDs. Query processing typically involves traversing links and paths (concatenation of links) through these sources. We model the data objects in these sources and the links between objects as an object graph. Analogous to database cost models, we use samples and statistics from the object graph to develop a framework to estimate the result size for a query on the object graph.

1 Querying Interlinked Sources

An abundance of biological data sources contain data about scientific entities, such as genes and sequences. Logical relationships between scientific objects are implemented as *links* between data sources. Scientists are interested in exploring these relationships between scientific objects, e.g., genes and bibliographic citations. Consider the query *"Return all citations of* PUBMED *that are linked to an* OMIM *entry that is related to some disease or condition."* To answer such queries, biologists and query engines alike must fully traverse links and paths (informally concatenations of links) through these sources given some start object in OMIM. Figure 1 illustrates the source graph for four data sources at the National Center for Biotechnology Information (NCBI). A scientist may choose the OMIM source, which contains information related to human genetic diseases, as a starting point for her exploration and wish to eventually retrieve citations from the PUBMED source. Starting with a keyword search on a certain disease, she can explore direct links between genes in OMIM and citations in PUBMED. She

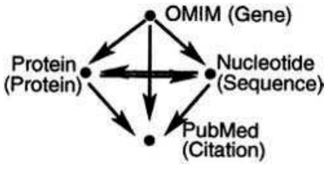


Fig.1. Source graph for NCBI data sources (and corresponding scientific entities)

can also traverse paths that are implemented using additional intermediate sources to learn about this relationship. In all, there are five paths (without loops) starting from OMIM and terminating in PUBMED. These paths are shown in Fig. 2.

(P1) OMIM → PUBMED
(P2) OMIM → NUCLEOTIDE → PUBMED
(P3) OMIM → PROTEIN → PUBMED
(P4) OMIM → NUCLEOTIDE → PROTEIN → PUBMED
(P5) OMIM → PROTEIN → NUCLEOTIDE → PUBMED

Fig. 2. All five paths from OMIM to PUBMED through the source graph of Fig. 1

The choice of paths has an impact on the result. For example, traversing a path via the PROTEIN source might yield less and different citations compared to a path via the NUCLEOTIDE source. This depends on the intermediate sources and corresponding entity classes that are traversed in a path, the contents of each source, the contents of each source link, etc.

These properties of paths and their effects are of interest from a number of perspectives: From a *query evaluation* viewpoint, one can estimate the cost and benefit of evaluating a query given some specific sources and paths. A second perspective that can profit from this work is that of *data curation*: Administrators of cross-linked data sources are interested in providing not only correct data, but also complete and consistent links to related data items in other data sources. Finally, the properties presented in this paper uncover *semantics* of the data sources and links between sources. Consider the five alternative paths in Fig. 2. Each of these paths yields a different number of distinct objects. Ordering these paths based on the cardinality of objects in PUBMED or comparing the overlap of objects among these alternate paths correspond to useful semantics that the researcher can exploit.

In this paper, we develop a model for the source graph while paying attention to properties of links and paths and properties of alternative links and paths. These properties of the source graph allow us to estimate properties of the result graph, i.e., the graph generated as a response to a query against the object graph. The approach is analogous to database cost models, where statistics of the database instance are used to predict the result cardinality for a query.

We consider four data sources from NCBI and the statistics of the corresponding object graph. We sample data from these sources to construct some results graphs, and we validate the accuracy of our framework to estimate the properties of the result graph. Together with related work in [10], our research provides a foundation for querying and exploring data sources.

There has been prior research on providing access to life science sources [2,3,7,14]. Example systems include DiscoveryLink [6], Kleisli and its successors [1], SRS [4], and Tambis [12]. Recent research in [11] has considered multiple alternate paths through sources but they have not addressed the properties of paths. In [8] Kleinberg et al. are interested in distinguishing characteristic shapes and connectivity in graphs but not in estimating the number of objects reached, as is our interest.

Properties of links and paths have been studied in the context of XML document processing in the XSketch project [13]. Given an XML document and the corresponding graph, the authors consider label-split graphs and backwards/forwards bi-similar graphs to obtain a synopsis of the original graph. The objective is a compact but accurate synopsis. Assuming statistical independence and uniform distribution, they determine the selectivity for complex path expressions using such synopses. Like us, they use these synopses in an estimation framework. Their approach differs from our approach in that we use statistics such as cardinality and average outdegree from the object graph, rather than detailed synopses.

2 Definitions

This section describes our model of the world and the data within. For formal definitions see [9]. In short, a logical graph *LG* with scientific entities as nodes is an abstraction (or schema) of the source graph *SG* with data sources as nodes. In turn, the object graph *OG* is an instance of *SG.* Finally, the result graph *RG* is a subset of *OG* and contains the data objects and links specific to a particular query. *LG, OG,* and *RG* are (somewhat) analogous to the schema, database instance, and result of a query. For simplicity of notation, we assume that a source provides data for a single scientific entity class. Thus, in analogy to databases, a source acts as a table. If a real world source provides data for more than one class, we model it as an individual source for each of its classes.

The object graph *OG* represents our model of all the objects and links that we consider. Each object is an instance of a particular class and each link is between two objects of different classes. A data object can have multiple outgoing and incoming links, not all objects have incoming or outgoing object links, and thus the object graph *OG* is not necessarily connected. Please note that there may be many real links in the sources that are not represented in our model, e.g., a data object could have a link to another data object in the *same* source.

In related work [10], we have defined a regular expression based query language over the entity classes of *LG. A* regular expression is satisfied by a set of result paths. Each path is a subset of data objects and object links from *OG.* The actual construction of the result graph with a set of real world databases is described in more detail in Sec. 4. These graphs were used to test our model.

3 Characterizing the Source Graph

To further our goal of supporting queries on life sciences data sources, we introduce our framework of properties of the source graph such as outdegree, result cardinality, etc. The framework uses statistics from the object graph *OG* such as source cardinality, link cardinality, etc.

Node and Link cardinality. The number of objects stored at a source and their link structure to other sources are among the most basic metadata to obtain, either from the administrators of the sources themselves, or by analyzing source samples. We define node and link cardinality for any graph *G* (for formal definitions, we refer to [9]). Then we apply these definitions to object graphs and result graphs as defined in the previous section.

We denote the cardinality of source *S* as $c^{OG}(S)$, and the estimated cardinality as $c^{OG}_{est}(S)$. We denote the number of links (link cardinality) between sources S_i and S_j as $l^{OG}(S_{i,j})$. A useful derived property is the average number of outgoing links from data objects, calculated along a path[4] p as $l_{out}(S^p_{i,i+1}) = l^{OG}(S^p_{i,i+1})/c^{OG}(S_i)$. The link image of a source is the set of data objects that are reachable in its implementation. Its cardinality is denoted as $l_{im}(S_{i,j})$. We are interested in the size of the link image, because this metadata improves the accuracy of our estimations. For brevity, we omit the path index p where the belonging of a source to a path is obvious.

Estimating result cardinality. Let m_1 be the number of starting objects found in source S_1. Following a given path p through sources $S_1, \ldots, S_n$, we construct the result path *RP* and estimate the number of distinct objects reached at the last source of the path, i.e., the result cardinality. To consider overlap of object links, we must determine the likely number of *distinct* objects found in S_i, if randomly choosing m objects from all $c^{OG}(S_i)$ objects in S_i. The probability to find exactly x distinct objects when picking m times from a set of $c^{OG}(S_i)$ objects in a source is (see [5])

$$\frac{\binom{c^{OG}(S_i)}{x} \cdot \binom{m-1}{m-x}}{\binom{m+c^{OG}(S_i)-1}{m}}. \tag{1}$$

For notational simplicity, we define m_i to be the expected number of links from source S_{i-1} to S_i, i.e., $m_i := c^{RP}_{est}(S_{i-1}) \cdot l^{RP}_{out}(S_{i-1,i})$ for $i > 1$. The expected number of distinct objects found in a source is the sum of all possible outcomes x multiplied with their probability from (1):

$$c^{RP}_{est}(S_i) = \begin{cases} m_1, & \text{if } i = 1; \\ \sum_{x=1}^{m_i} x \cdot \frac{\binom{l_{im}(S_{i-1},S_i)}{x} \cdot \binom{m_i-1}{m_i-x}}{\binom{m_i+l_{im}(S_{i-1},S_i)-1}{m_i}} & \text{if } i > 1. \end{cases} \tag{2}$$

In this formula, we must recursively replace the input value m_i with the number of distinct objects found in the previous source along the path. This calculation makes the simplifying assumption of link independence along a path. Informally,

[4] We use path in the usual graph theory sense, i.e., a set of successive directed links through the object graph.

we assume that the probability of a link from some object in source S_{i-1} to an object o in source S_i is independent of the probability of a link from object o in S_i to an object in source S_{i+1}. Future work will examine different dependency cases among links, such as containment, disjointness, etc.

4 Validating the Framework

We report on an experiment on data sources of the National Center for Biotechnology Information (NCBI) to illustrate that querying well-curated sources managed by a single organization may result in different semantics, depending on the specific link, path, and intermediate sources that are chosen. Our experiment was limited to the source graph described in Fig. 1. Data was sampled from each of the sources to construct several results graphs *RG*. Validation involved comparing measured values of the *RG*s with our estimates.

Creating Samples and Measurements. The methodology to create sample result graphs corresponds to retrieving bibliographical references from PUBMED that are linked to genes relevant to a given disease or medical condition. We fully explore all links and paths that exist between objects in the four sources, given the start set of objects in OMIM. The study focused on three medical conditions: *cancer, aging,* and *diabetes.* A list of relevant keywords for each condition was used to retrieve relevant genes from OMIM. These genes constitute the starting set of objects. We created 12 result graphs, 4 for each of the conditions. Each result graph contains a collection of 140 to 150 OMIM records and usually many more objects from the other sources.

Figure 3 (left) shows the results of one such experiment for the condition *aging,* starting with 141 OMIM records along with the *measured* values for different paths through the result graph. Each edge label shows the link cardinality and each node label shows the number of distinct objects found by following those links (node cardinality).

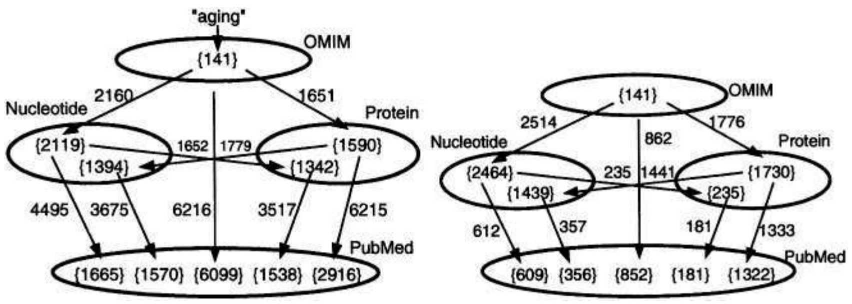


Fig. 3. The result graph from experiments on aging (left) and calculation (right)

Estimations. As an input to our formulas, we obtained statistics on the object graph *OG* for February 2003 from NCBI. These statistics include node cardinality for each of the sources and link cardinality for any pair of sources. The statistics were input to Formula (2) to estimate the number of distinct objects found at each node. Figure 3 (right) shows the results of these calculations. The number of OMIM entries (141) in the start node was chosen to exactly correspond to the result graph *RG* of Fig. 3 (left). The node labels on the right give the estimated number of distinct objects encountered along a path and edge labels give the estimated number of links.

Comparison. We now compare the measurements and estimations of the in Fig. 3. To understand the discrepancies, consider Tab. 1. For each link in the five paths, we report the number of measured links (`LinkMeas`), the number of estimated links (`LinkEst`), and the error in estimation as the ratio of estimation and measurement (`LinkEst/LinkMeas`).

Link	LinkMeas	LinkEst	ERROR = LinkEst/LinkMeas	ObjMeas	ObjEst	ERROR = ObjEst/ObjMeas
Om-Pu	6,216	862	0.139	6,099	852	0.140
Om-Nu	2,160	2,514	1.164	2,119	2,464	1.163
Om-Pr	1,651	1,776	1.076	1,590	1,730	1.088
(Om-)Nu-Pu	4,495	612	0.136	1,665	609	0.366
(Om-)Pr-Pu	6,215	1,333	0.214	2,916	1,322	0.453
(Om-)Nu-Pr	1,652	235	0.142	1,342	235	0.175
(Om-)Pr-Nu	1,779	1,441	0.810	1,394	1,439	1.032
(Om-Nu-)Pr-Pu	3517	181	0.051	1,538	180	0.117
(Om-Pr-)Nu-Pu	3,675	357	0.097	1,570	356	0.227

Table 1. Fractional error in estimation for "aging"

For those links where the error fraction for both links and objects is close to 1.0 (low error), what appears common is that the number of distinct objects is in the same range as the number of links. The independence assumption for links (objects) of our model appears to be upheld here. However, for the rest of the links (objects) where the error fraction is close to 0.0 (high error) indicates that the assumption of an uniform distribution with independence among links (objects) is not supported.

5 Training and Testing

Having twelve result graphs *RG,* one for each set of OMIM starting objects, we enhanced our estimations using a training and testing technique. That is, we used all but one of the result graphs to gain insight into expected path cardinalities given certain input parameters (training). The single remaining result graph served as the test data set. Through this training, we are able to overcome the independence assumption made in Sec. 3 for result cardinality estimation.

Model for Training. For result graphs *RG* we present some additional notation and an expression for our estimation. For formal definitions, please see [9]. Link participation $l_{par}^{RG}(S_{i,j})$ is the number of objects in S_i in *RG* having at least one outgoing link to an object in S_j. Link outdegree in *RG* using participation (instead of the entire set of object in S_i) is denoted $l_{out}^{RG'}(S_{i,j})$ and describes the average number of links of each data object in S_i in *RG* pointing to an object of source S_j in *RG.* Along a path p in *RG,* average outdegree based on participation is calculated as $l_{out}^{RG'}(S_{i,i+1}) = l_{out}^{RG}(S_{i,i+1})/l_{par}^{RG}(S_{i,i+1})$. To overcome the independence assumption made earlier, we define the *path dependence factor pdf* capturing the statistics from the *RG*s of an object in S_i having both an inlink from S_{i-1} and an outlink to an object in S_{i+1}: $pdf(S_i) := l_{par}^{RG}(S_{i,i+1})/l_{im}^{RG}(S_{i-1,i})$. We further define the *duplication factor df* to capture the statistics from the *RG* of two links from S_i pointing to the same object in S_{i+1}: $df(S_{i,i+1}) := l_{im}^{RG}(S_{i,i+1})/l_{out}^{RG}(S_{i,i+1})$.

Estimating result cardinality. Following a given path p through sources $S_1, \ldots, S_n$, we construct the result path *RP.* Let m_1 be the number of participating objects found in source S_1. Using $pdf(S_i)$ and average outdegree based on participation $l_{out}^{RG2}(S_{i,i+1})$, we can estimate the object cardinality as follows:

$$\begin{aligned} c_{est}^{RP}(S_k) = m_1 \cdot l_{out}^{RG'}(S_{1,2}) \cdot df(S_{1,2}) \\ \cdot \Pi_{i=2,\ldots,k-1} \left[pdf(S_i) \cdot l_{out}^{RG'}(S_{i,i+1}) \cdot df(S_{i,i+1}) \right], \ k > 2 \qquad (3) \end{aligned}$$

Validating the Model. For each of the 12 sampled object graphs (see Sec. 4), statistics, such as path dependence factor, duplication factor, average outdegree, etc., were calculated. We then chose one object graph, namely aging1, to make predictions on by using the average value taken over the remaining 11 objects graphs. For aging1, the average was calculated over aging2 through diabetes4. The result graph for aging1 is shown in Fig. 4.

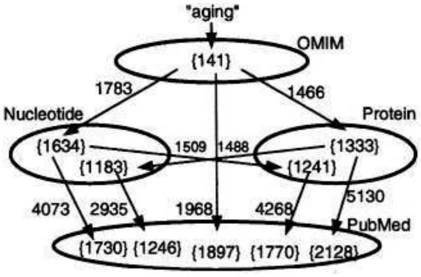


Fig. 4. The result graph from predictions on aging1

The predictions are compared with the values from experiments and the results are shown in Tab. 2. The table is similar to Tab. 1, where we tabulate errors in estimation assuming independence of links. Three of the links (Om-Nu-Pu, Om-Nu-Pr, and Om-Nu-Pr-Pu) have good predictions, five of them have a moderate prediction, and the only poor prediction is for the link Om-Pu.

"aging1" Link	LinkMeas	LinkEst	ERROR = LinkEst/LinkMeas	ObjMeas	ObjEst	ERROR = ObjEst/ObjMeas
Om-Pu	6,216	1,968	0.317	6,099	1,897	0.311
Om-Nu	2,160	1,783	0.825	2,119	1,634	0.771
Om-Pr	1,651	1,466	0.888	1,590	1,333	0.838
(Om-)Nu-Pu	4,495	4073	0.906	1,665	1730	1.039
(Om-)Pr-Pu	6,215	5,130	0.825	2,916	2,128	0.730
(Om-)Nu-Pr	1,652	1509	0.913	1,342	1241	0.925
(Om-)Pr-Nu	1,779	1,488	0.836	1,394	1,183	0.849
(Om-Nu-)Pr-Pu	3517	4268	1.214	1,538	1770	1.151
(Om-Pr-)Nu-Pu	3,675	2,935	0.799	1,570	1,246	0.794

Table 2. Fractional errors in prediction for aging1

6 Conclusions

The presented research is only a starting point of understanding Web-based life sciences sources and their relationships with one another. Future work concentrates both on the extension and generalization of the set of properties and on the usage of the presented properties for different scenarios. Additionally, we plan to extend our model by allowing other distributions of links (stored as histograms), by including multiple sources for individual scientific entities, and by considering more complex link structures, including cycles and loops. Together with results presented in [10], this application area promises biologists the ability to efficiently and effectively query interlinked data sources, such as those at NCBI.

Acknowledgements. This research is partially supported by NSF grants IIS-0219909, EIA0130422 and IIS0222847, and the NIH National Institutes of Aging Grant 1 R03 AG21671-01. We thank Barbara Eckman of IBM Life Sciences, David Lipman and Alex Lash of NCBI, Damayanti Gupta, Marta Janer, and Michael Jazwinski.

References

1. S. Davidson, J. Cabtree, B. Brunk, J.Schug, V. Tannen, C. Overton, and C. Stoeckert. K2/Kleisli and GUS: Experiments in integrated access to genomic data sources. *IBM Systems Journal,* 40(2), 2001.
2. B. Eckman, A. Kosky, and L. Laroco. Extending traditional query-based integration approaches for functional characterization of post-genomic data. *BioInformatics,* 17(2), 2000.
3. B. Eckman, Z. Lacroix, and L. Raschid. Optimized seamless integration of biomolecular data. *Proc. of the IEEE Int. Symp. on Bio-Informatics and Biomedical Engineering,* 2001.
4. T. Etzold and P. Argos. SRS: An indexing and retrieval tool for flat file data libraries. *Computer Applications of Biosciences,* 9(1), 1993.
5. W. Feller. *An Introduction to Probability Theory and Its Applications.* John Wiley & Sons, New York, NY, 1968.

6. L. Haas, P. Kodali, J. Rice, P. Schwarz, and W. Swope. Integrating life sciences data - with a little Garlic. *Proc. of the IEEE Int. Symp. on Bio-Informatics and Biomedical Engineering,* 2000.
7. G. Kemp, C. Robertson, and P. Gray. Efficient access to biological databases using CORBA. *CCP11 Newsletter,* 3.1(7), 1999.
8. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *ACM Computing Surveys,* 46(5):604–632, 1999.
9. Zoé Lacroix, Hyma Murthy, Felix Naumann, and Louiqa Raschid. Links and paths through life sciences data sources. Technical Report, Humboldt-Universität zu Berlin, Institut für Informatik, 2004.
10. Zoé Lacroix, Louiqa Raschid, and Maria-Esther Vidal. Efficient techniques to explore and rank paths in life science data sources. In *Proc. of the Int. Workshop on Data Integration for the Life Sciences (DILS),* Leipzig, Germany, 2004.
11. P. Mork, R. Shaker, A. Halevy, and P. Tarczy-Hornoch. PQL: A declarative query language over dynamic biological data. *Proc. of the AMIA,* 2002.
12. N.W. Paton, R. Stevens, P.G. Baker, C.A. Goble, S. Bechhofer, and Brass. Query processing in the tambis bioinformatics source integration system. *Proc. of the IEEE Intl. Conf. on Scientific and Statistical Databases (SSDBM),* 1999.
13. N. Polyzotis and M. Garofalakis. Structure and value synopses for XML data graphs. *Proc. of the Conf. on Very Large Databases (VLDB),* 2002.
14. T. Topaloglou, A. Kosky, and V. Markovitz. Seamless integration of biological applications within a database framework. *Proc. of the Intl. Conf. on Intelligent Systems for Molecular Biology (ISMB),* 1999.

Pathway and Protein Interaction Data: from XML to FDM Database

Graham J.L. Kemp and Selpi

Department of Computing Science, Chalmers University of Technology,
SE-412 96 Göteborg, Sweden

Abstract. This paper describes our experience with the first steps towards integrating pathway and protein interaction data with other data sets within the framework of a federated database system based on the functional data model. We have made use of DTD and XML files produced by the BIND project. The DTD provides a specification for information about biomolecular interactions, complexes and pathways, and can be translated semi-automatically to a database schema. The load utility uses metadata derived from this schema to help identify data items of interest when recursively traversing a Prolog tree structure representing the XML data. We also show how derived functions can be used to make explicit those relationships that are present in data sets but which are not fully described in DTD files.

1 Introduction

In recent years there has been a rapid expansion in the quantity and variety of biological data available to researchers. These include data on protein and genome sequences and structure, gene and protein expression, molecular interactions and biological pathways. Scientists' ability to use these data resources effectively to explore hypotheses *in silico* is enhanced if it is easy to ask precise and complex questions that span across several different kinds of data resources in order to find the answer. In our earlier work we have built a federated system in which queries requiring data values from distributed heterogeneous data resources are processed by a prototype program called the P/FDM Mediator [8], which is based on the P/FDM object database system [6]. Tasks performed by the P/FDM Mediator include determining which external databases are relevant in answering users' queries, dividing queries into parts that will be sent to different external databases, translating these subqueries into the language(s) of the external databases, and combining the results for presentation.

A first step in adding a new data resource to our federation is to describe the contents of the new data resource using the functional data model (FDM) [10], which is an example of a semantic data model. Data resources including SRS [5] and ACEDB [4] have been mapped in this way in earlier work. Once this has been done code generators within the P/FDM mediator can produce *ad hoc* queries in the query language used by these systems and these can be dispatched to the remote systems for execution.

Since there wasn't an existing pathway resource to which *ad hoc* queries could be submitted we undertook to load data from XML files provided by another project into a database management system which could ultimately be used as a data resource within a federated system. We decided to use the P/FDM database for storing pathway and interaction data in preference to a database based on the relational model. This was because the XML files are "non-relational", for example these may contain examples of multi-valued attributes and relationships which can be modelled directly using the FDM, but which require a more complex mapping if using the relational model. Further, given the graph structure of interaction networks and the hierarchical part-subpart structure of biomolecular complexes we anticipated being able to benefit from the ability to express recursive queries in Daplex, the FDM's data definition language and query language, against these naturally recursive structures in the data.

In Section 2 we briefly describe some of the candidate pathway data resources that we considered using in this work, and the reasons why BIND was selected. We describe how we generated a functional data model schema from BIND's DTD and how XML files containing pathways, interactions and biomolecular complexes were loaded into the P/FDM database management system in Section 3. In Section 4 we show how the resulting database is queried. Finally, we reflect on our experience in this project, and the lessons that can be learned in designing a data exchange format for pathway and interaction data.

2 Biological Pathway and Interaction Data Sources

Several data resources with information on biological pathways and interactions are under development. These include BIND [2], KEGG [7], MIPS [9] and BioPAX[1]. These differ in the quantity and variety of data available, the file formats used for data distribution, the design of their metadata, and their state of development.

The BIND project [2, 1] has produced a data specification for information about biomolecular interactions, complexes and pathways. This specification was initially defined in ASN.1 [3]. However, tools provided by NCBI have been used to transform this description into an XML DTD[2]. Similarly, data files conforming to the ASN.1 specification have been transformed systematically into XML. Both ASN.1 and XML versions of data files are available from the BIND web site.

KEGG [7] is another biochemical pathway resource. Data from this project are available in KGML (KEGG Markup Language) format and, recently, in XML. However, XML tags have been designed differently in BIND and KEGG. In KEGG the focus is on the graphical presentation of pathway diagrams and the XML tags include layout and presentation elements, whereas in BIND the XML tags relate more directly to biological concepts. BIND and KEGG also use different conventions for naming attributes in their XML DTDs.

[1] http://www.biopax.org/

[2] http://www.ncbi.nih.gov/IEB/ToolBox/XML/ncbixml.txt

The MIPS Comprehensive Yeast Genome Database [9] includes detailed data on protein-protein interactions, pathways and complexes in yeast. However, this resource does not currently provide data in an XML format, and the project web site does not include a description of the resource's metadata.

The Biological Pathways Exchange project (BioPAX) aims to produce a common exchange format for biological pathway data. That project uses frame-like structures with classes and slots for describing pathways. This relatively new project is at the draft release stage, and no data are currently available in this format.

We decided to use data from BIND in our work since this is available in an XML format, and is accompanied by a description in XML DTD that defines tags that are based on biological concepts. Further, since the DTD file is generated in a systematic way from an ASN.1 specification, the entity and attribute names have a consistent form.

3 Implementing an FDM Database from BIND's DTD and XML Files

Our implementation consists of two main software components: (i) a Perl program that reads BIND XML DTD and produces Daplex data definition statements, and (ii) an XML load utility that can load data from XML data files into the P/FDM database.

First we shall consider the task of producing the Daplex schema. Entity types can be recognised as those elements defined the DTD file that have names that do not contain an underscore character and have at least one non-optional attribute. There are two entity types in the extract from the BIND DTD in Figure 1: BIND-Interaction and BIND-object. Scalar attributes and relationships are also identified from the DTD file. Daplex data definition statements generated for this DTD fragment are shown in Figure 2, and these are compiled by the P/FDM system into metadata that are stored in the database.

A serious difficulty when trying to produce a database schema automatically from a DTD file is that the DTD does not contain information about which attributes constitute the keys of the entity classes. We anticipated having to specify these manually, however for many classes a satisfactory candidate key can be formed by taking the union of all non-optional attributes and relationships defined on that class in the DTD. In some cases (including BIND-Interaction) this simple approach produces a super-key from which we have to remove attributes and relationships to form a candidate key. For some others (including BIND_object) the set of non-optional attributes and relationships is insufficient to uniquely identify instances of the class and so one or more optional attributes need to be added to form a candidate key. In one case it was necessary to remove a non-optional relationship and add one optional attribute to form a candidate key. However, for the majority of cases this simple approach yielded a satisfactory key automatically and fewer than ten keys definitions had to be adjusted by hand.

```
<!ELEMENT BIND-Interaction (
                BIND-Interaction_iid ,
                BIND-Interaction_a ,
                BIND-Interaction_b ,
                BIND-Interaction_priv? )>

<!ELEMENT BIND-Interaction_iid ( Interaction-id )>
<!ELEMENT BIND-Interaction_a ( BIND-object )>
<!ELEMENT BIND-Interaction_b ( BIND-object )>
<!ELEMENT BIND-Interaction_priv %BOOLEAN; >
<!ATTLIST BIND-Interaction_priv value ( true | false ) "false" >

<!ELEMENT Interaction-id ( %INTEGER; )>

<!ELEMENT BIND-object (
                BIND-object_short-label ,
                BIND-object_descr? ,
                BIND-object_user-id? )>

<!ELEMENT BIND-object_short-label ( #PCDATA )>
<!ELEMENT BIND-object_descr ( #PCDATA )>
<!ELEMENT BIND-object_user-id ( %INTEGER; )>
```

Fig. 1. An extract from BIND's DTD. Several elements have been omitted to make the figure concise, while still illustrating features described in the text.

```
declare BIND_Interaction ->> entity
declare BIND_object ->> entity

declare iid(BIND_Interaction) -> integer
declare a(BIND_Interaction) -> BIND_object
declare b(BIND_Interaction) -> BIND_object
declare priv(BIND_Interaction) -> boolean

declare short_label(BIND_object) -> string
declare descr(BIND_object) -> string
declare user_id(BIND_object) -> integer

key_of BIND_Interaction is iid
key_of BIND_object is short_label, descr
```

Fig. 2. Daplex data definition statements generated from the DTD fragment shown in Figure 1.

```
<BIND_Interaction>
  <BIND_Interaction-iid>
    <Interaction_id>118</Interaction_id>
  </BIND_Interaction-iid>
  <BIND_Interaction-a>
    <BIND_object>
      <BIND_object-short_label>EGF_EGFR complex</BIND_object-short_label>
      <BIND_object-descr>Epidermal Growth Factor (EGF) bound to Epidermal
Growth Factor Receptor (EGFR)</BIND_object-descr>
      <BIND_object-user_id>0</BIND_object-user_id>
    </BIND_object>
  </BIND_Interaction-a>
  <BIND_Interaction-priv value=""false""/>
</BIND_Interaction>
```

Fig. 3. An extract from the BIND XML data file for interaction number 118.

The XML load utility has been implemented in Prolog — the language in which most of the P/FDM database management system is written. Figure 3 illustrates the structure of data in a BIND XML file. An XML document can be parsed and compiled into a nested Prolog term structure which has the same tree structure as the original XML document. The XML load utility then recursively traverses this tree to find data items that will be loaded into the database. This program first identifies the type of the top-level element in the XML file (near the root of the tree), and retrieves the metadata entry for this type from the P/FDM database. The names of the key attributes for this type are extracted from the metadata entry, and the Prolog program then searches through the tree to find values for these attributes. If values can be found for all of these then a new entity instance is created using the P/FDM update predicate *newentity*. The first two arguments in the call to *newentity* are the entity type name and a list of values for the key attributes. The third argument becomes instantiated to the internal identifier of the entity instance created by the call (e.g. BIND_Interaction(10) in the first call in Figure 4).

The XML load utility then proceeds to look for values for any other attributes and relationships defined on this type. When values are found these are added to the database using the P/FDM update predicate *addfnval* (add function value), which takes the attribute name, the internal identifier of the entity instance and the value of the attribute as its three arguments.

When a relationship is present in the XML data file the load utility must use *addfnval* giving the internal identifier of the related entity instance as the value of the relationship function. If this entity instance is already in the database then this function value can be added immediately. Otherwise, it is necessary to create the related entity instance and add it to the database first, before the relationship function is added. An example of this can be seen in Figure 4, where the BIND object with the key components “EGF_EGFR complex” and “Epidermal Growth Factor (EGF) bound to Epidermal Growth Factor Receptor

```
newentity('BIND_Interaction', [118], 'BIND_Interaction'(10)),
newentity('BIND_object', ['EGF_EGFR complex','Epidermal Growth Factor(EGF)
 bound to Epidermal Growth Factor Receptor (EGFR)'], 'BIND_object'(18)),
addfnval(user_id, ['BIND_object'(18)], 0),
addfnval(a, ['BIND_Interaction'(10)], 'BIND_object'(18)),
addfnval(priv, ['BIND_Interaction'(10)], false)
```

Fig. 4. Instantiated Prolog update goals corresponding to the XML data in Figure 3.

(EGFR)" has to be created before a value for relationship function a is added to the database.

We have loaded into P/FDM all BIND pathways that were available in an XML format, together with all related BIND objects and BIND interactions. The XML load utility does not attempt to load all of the data items contained in the original XML documents. Rather, it searches only for data items corresponding to attributes and relationships declared in the schema. Any other data items or elements in the XML data are ignored by the load utility.

4 Querying the Database

The definitions of several derived relationship functions are shown in Figure 5. It is useful to be able to define such functions so that implicit relationships in the database can be made explicit. For example, the BIND Interactions that are related to a BIND Molecular Complex are represented in the XML version as a list of integers that correspond to interaction identifiers, implicitly defining a multi-valued relationship between BIND Molecular Complexes and BIND Interactions. The function definition for *complex_interactions* in Figure 5 makes this relationship explicit. Function *all_complex_subcomplexes* is a recursive function that finds all BIND Molecular Complexes that are part of other BIND Molecular Complexes. This function is used in the Daplex query in Figure 5.

5 Discussion and Conclusions

Providing data in an XML format, even when accompanied by a DTD, is not sufficient to ensure that it will be easy to import these data automatically into a database management system. To facilitate this, it is important that the XML tags are well designed. The systematic naming of DTD elements in BIND is very useful for this purpose. However, it would be helpful if key attributes were declared explicitly in the DTD.

While we have been able to capture most of the information present in the DTD files, we have not been able to capture it all. In particular, we do not map those attributes whose value might be one of several different types since a function in P/FDM must return values of a single type.

Daplex function definitions:

```
define interaction_objects(i in BIND_Interaction)
       ->> BIND_object
  {a(i), b(i)};


define complex_interactions(c in BIND_Molecular_Complex)
       ->> BIND_Interaction
  i in BIND_Interaction such that iid(i) in interaction_list(c);


define complex_objects(c in BIND_Molecular_Complex)
       ->> BIND_object
  interaction_objects(complex_interactions(c));


define complex_subcomplexes(c in BIND_Molecular_Complex)
       ->> BIND_Molecular_Complex
  s in BIND_Molecular_Complex such that
    descr(s) in short_label(complex_objects(c));


define all_complex_subcomplexes(c in BIND_Molecular_Complex)
       ->> BIND_Molecular_Complex
  ( complex_subcomplexes(c)
  union
    all_complex_subcomplexes(complex_subcomplexes(c)) );
```

Daplex query:

```
for each c in BIND_Molecular_Complex
  for each s in all_complex_subcomplexes(c)
    print(mcid(c), descr(c), mcid(s), descr(s));
```

Query results:

```
12971 (EGF_EGFR) dimer complex bound to ATP 12970 (EGF_EGFR) dimer complex
12971 (EGF_EGFR) dimer complex bound to ATP 12966 EGF_EGFR complex
12970 (EGF_EGFR) dimer complex              12966 EGF_EGFR complex
```

Fig. 5. Function definitions and a query.

Derived functions in the functional data model, such as those shown in Figure 5, can make explicit those relationships that are present but which are not fully described in a DTD file. These functions make it more convenient to express queries involving related objects.

The XML load utility is general, and it should be possible to use this with other XML data sets. The Perl program, however, is specific to the style of DTD files produced by the NCBI tools and would have to be modified before it can be used with other DTDs.

The work described in this paper demonstrates that a database schema can be produced from an XML DTD file with only a little manual intervention, and data from XML files can then be loaded into a database management system within which it can be explored using an *ad hoc* query language. This represents a useful first step towards integrating pathway and protein interaction data with other data stored in the P/FDM database management system, or with other data sets accessible within a federated architecture in which a P/FDM database is a constituent data resource.

Acknowledgements

We are grateful for a scholarship from the Swedish Foundation for International Cooperation in Research and Higher Education (S.).

References

[1] Gary D. Bader, Doron Betel, and Christopher W.V. Hogue. BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Research,* 31:248–250, 2003.

[2] Gary D. Bader and Christopher W.V. Hogue. BIND – a data specification for storing and describing biomolecular interactions, molecular complexes and pathways. *Bioinformatics,* 16:465–477, 2000.

[3] O. Dubuisson. *ASN.I Communication Between Heterogeneous Systems.* Morgan Kaufmann Publishers, 2000.

[4] R. Durbin and J. Thierry-Mieg. *Syntactic Definitions for the ACEDB Data Base Manager,* 1992.

[5] T. Etzold and P. Argos. SRS an indexing and retrieval tool for flat file data libraries. *CABIOS,* 9:49–57, 1993.

[6] P.M.D. Gray, K.G. Kulkarni, and N.W. Paton. *Object-Oriented Databases: a Semantic Data Model Approach.* Prentice Hall Series in Computer Science. Prentice Hall Int. Ltd., 1992.

[7] M. Kanehisa and S. Goto. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research,* 28:27–30, 2000.

[8] G. J. L. Kemp, N. Angelopoulos, and P. M. D. Gray. Architecture of a Mediator for a Bioinformatics Database Federation. *IEEE Transactions on Information Technology in Biomedicine,* 6:116–122, 2002.

[9] H.W. Mewes, D. Frishman, U. Güldener, G. Mannhaupt, K. Mayer, M. Mokrejs, B. Morgenstern, M. Münsterkötter, S. Rudd, and B. Weil. MIPS: a database for genomes and protein sequences. *Nucleic Acids Research,* 28:31–34, 2002.

[10] D.W. Shipman. The Functional Data Model and the Data Language DAPLEX. *ACM Transactions on Database Systems,* 6(1):140–173, 1981.

Author Index