### Towards transparent access to multiple biological databanks

#### Patrick Lambrix, Vaida Jakonienė

Department of Computer and Information Science Linköpings universitet, Sweden patla@ida.liu.se

#### Abstract

Nowadays, biologists use a number of large biological databanks to find relevant information for their research. Users of these databanks face a number of problems. One problem is that users are required to have good knowledge about the contents, implementations and conceptual models of many databanks to be able to ask precise and relevant questions. Further, the terminology that is used in the different databanks may be different. Also, when asking complex queries to multiple databanks, users need to construct a query plan on their own possibly leading to poor performance or not even obtaining results. To alleviate these problems we define an architecture for systems that deal with these problems by allowing for a transparent and integrated way to query the multiple sources. The contribution of this paper is threefold. First, we describe a study of current biological databanks. Then, we propose a base query language that contains operators that should be present in any query language for biological databanks. Further, we present an architecture for a system supporting such a language and providing integrated access to the highly distributed and heterogeneous environment of biological databanks.

#### 1 Introduction

Nowadays, biologists use a number of large biological databanks such as SWISS-PROT (Bairoch and Apweiler 2000), EMBL (Stoesser et al. 2001) and Genbank (Benson et al. 2000) to find relevant information for their research. An example of this is the field of comparative genomics where information on the location and functionality of genes and the interaction between genes is derived by doing sequence comparison with sequences in the databanks for which there already exist established results. There are a huge amount of such biological databanks. For instance, the online database issue and collection 2001 of the Nucleic Acids Research journal (NAR 2001) describes briefly 95 information sources and contains a Molecular Biology Database Collection with 281 information sources. DBCAT (DBCAT -) lists 511 sources.

One of the areas in the field of bioinformatics concerns the study of the creation, maintenance and use of these biological databanks. Users of these databanks face a number of problems. This is partly due to the inherent complexity of building such databanks. Some of the facts that contribute to this complexity are, for instance, the fact that biological data is highly complex when compared with data in most other domains, the amount and range of variability in data is high and the schemas in biological databanks change at rapid pace (Elmasri and Navathe 2000). However, another source of the problems is the ad hoc way in which some of these databanks have been built without much consideration for the practice in and lessons learned from building complex data sources in related areas such as distributed databases and information retrieval.

One problem that users face is that they are required to have good knowledge about which databanks contain which information as well as about the query languages or user interfaces and the conceptual models of many databanks. For example, there are 29 publicly accessible srs6 servers providing access to over 300 different databanks. It is possible to use systems such as Sequence Retrieval System (SRS -) to write queries involving the information in different databanks, but it is still required to select the databanks over which the queries should be performed.

Often there is a form-based user interface that helps the user partly, although different systems may use different ways of filling in the data fields in the forms. With respect to querying most databanks allow for full-text search and for search with predefined keywords. The structure of the databanks may be different (e.g. flat files, relational databases, objectoriented databases). Also the formats of the query results may be different. Biologists, however, should not need to have good knowledge about the specific implementations of the databanks, but instead a transparent solution should be provided.

Another problem is that representations of the same data by different biologists will likely be different. For instance, two databanks may store different information about the same entity. In figure 1 we show part of the Genelynx (GeneLynx -) entry for the adrenergic alpha-2B receptor, which is a hormone receptor for adrenaline. Genelynx collects hyperlinks for each human gene. The figure shows (part of) the links to databanks that have information about this receptor with the identifiers in the databanks in brackets. One of the databanks, SWISS-PROT (Bairoch and Apweiler 2000), contains information about the name, synonyms, source organism, taxonomy, references, function, subcellular location, similarity, features and amino-acid sequence. KEGG (KEGG -), on the other hand, contains information about the name, definition, class, position and aminoacid and nucleotide sequences. Also, the same entity may have different names in different databanks. This is the case, for instance, for entities for which a standardization committee has introduced a new name and both the new and the old names are used in (different) databanks. To solve this problem methods for representing and reasoning about biological data are needed. Also approaches for identifying the differences between conceptual models and integrating them are required.

Copyright ©2003, Australian Computer Society, Inc. This paper appeared at First Asia-Pacific Bioinformatics Conference (APBC2003), Adelaide, Australia. Conferences in Research and Practice in Information Technology, Vol. 19. Yi-Ping Phoebe Chen, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Genelynx #3309 Gene name: ADRA2B Description: adrenergic, alpha-2B-, receptor Summary pages: Unigene (Hs.247686), LocusLink (151), GeneCards (ADRA2B), Swiss-Prot (A2AB\_HUMAN), KEGG gene (151), EGAD (28203), euGenes (HUgn0000151), HumanPSD (ADRA2B) Genomic resources: NCBI/EBI/DDBJ (AF005900, M34041), GDB (120539), GenAtlas(ADRA2B)

Figure 1: Information about ADRA2B in different sources.

When users have a complex query where information from different sources needs to be combined, they often have to divide the queries into a number of simple queries, submit the simple queries to one databank at a time and combine the results themselves. This is a non-trivial task containing steps such as deciding which databanks to use and in which order, how terms in one databank map to terms in other databanks and how to combine the results. A mistake in any of these steps may lead to inefficient processing or even in not obtaining a result. A requirement for biological databanks is therefore that they allow for complex queries in a highly distributed and heterogeneous environment.

To alleviate these problems we want to define an architecture for systems that deal with these problems by allowing for a transparent and integrated way to query the multiple sources. The systems should provide a long-term solution taking into account the complexity of the field as well as the fact that there are a large number of legacy systems. Our approach can be seen as a view integrating approach (Davidson et al. 2001). In this kind of approach the underlying schemas of the resources are integrated into a global schema and this schema is queried in a highlevel query language. The contribution of this paper is threefold. First, we describe a study of a number of current biological databanks. The focus in the study was to investigate the query support of the different databanks. Then, we propose a high-level base query language that contains operators that should be present in any query language for multiple bio-logical databanks. The choice is based on the study of current practice as well as on a (currently small) number of interviews. In this paper we restrict the scope of the query language to text. We observe that the proposed query language is not necessarily used as an end-user query language. For the common end user the language is likely to be hidden behind a user interface. The study is described in section 2. The query language is introduced in section 3. In section 4 we propose an architecture for a system that supports this query language. The architecture is currently being implemented in a prototype. Related work is discussed in section 5 and the paper concludes in section 6.

#### 2 Study of databanks

We decided to study a number of the most used and best known databanks and chose GenBank (Benson et al. 2000), EMBL (Stoesser et al. 2001), DDBJ (DDBJ -), SWISS-PROT (Bairoch and Apweiler 2000), PIR (Wu et al. 2002), ENZYME (Bairoch 2000), PDB (Berman et al. 2000), MMDB (Wang et al. 2002), PROSITE (Falquet et al. 2002), PRINTS (Attwood 2002) and BLOCKS (Henikoff et al. 2000). For these systems we investigated the organization of the data, the data content and the data retrieval possibilities. With respect to the organization of the data

Databank	R	L	DB
GenBank	Y		EMBL, DDBJ
EMBL	Y		GenBank, DDBJ
DDBJ	Y		EMBL, GenBank
SWISS-PROT	Y	Y	GenBank, EMBL DDBJ
PIR	Y	Y	GenBank, EMBL DDBJ
ENZYME	Y	Υ	IUBMB
PDB	Y		
MMDB			PDB
PROSITE	Y	Υ	SWISS-PROT
PRINTS	Y		SWISS-PROT, TREMBL
BLOCKS			Interpro

Table 1: Source of the data. (Researchers (R), Literature (L), Other databanks (DB).)

Databank	FTP server	user interface
GenBank	genbank flat file,	genbank flat file
	asn.1	
EMBL	embl flat file	embl flat file,
		fasta, xml
DDBJ	ddbj flat file	ddbj flat file,
		embl flat file,
		xml, fasta
SWISS-PROT	swissprot flat file,	user friendly view
	fasta	swissprot flat file
PIR	nbfr-pir, codata,	nbfr-pir, codata,
	fasta, xml	fasta, xml
ENZYME	enzyme flat file,	user friendly view,
	asn.1	enzyme flat file
PDB	pdb flat file,	pdb flat file,
	mmCIF	mmCIF
MMDB	asn.1	mmdb flat file
PROSITE	prosite flat file	user friendly view,
		prosite flat file
PRINTS	prints flat file	prints flat file
BLOCKS	blocks flat file	blocks flat file

Table 2: Data formats.

we looked at the kind of data, the source, the data model, the update frequency and the location of the data. The chosen databanks store information about nucleotide sequences (GenBank, EMBL, DDBJ), protein sequences (SWISS-PROT, PIR, ENZYME), 3D macromolecular structures (PDB, MMDB) and protein families (PROSITE, PRINTS, BLOCKS). The source of the data can be researchers (that submit their data), literature (data from published articles) and other databanks. This is summarized in table 1. Data can usually be retrieved in different ways: via a web interface, ftp and e-mail. The underlying data models for these databanks are the flat file model, the relational model and the object-relational model. Also different formats may be used for the databank behind the web interface and the databank that can be loaded from ftp servers. A summary of the formats is given in table  $\overline{2}$ .

The content information can be grouped into the header, the annotation and the actual information. The header contains a unique identifier for the data item, one or more entry dates, one or more names, the source of the information and references. The annotation part contains comments and feature information. Finally, there is additional information that can contain a protein or DNA sequence, structure descriptions or experiment descriptions depending on the kind of databank. The level of detail of the data is also different.

With respect to retrieval capabilities we found that most databanks allow for queries based on the occurrence of text within a data item (full-text search) and all databanks support queries based on the occurrence of a text string within certain predefined fields. The user is often guided by the retrieval interface of the systems as to which fields are searchable. Two of the databanks, ENZYME and PRINTS, also allow for

Databank	UI	CL
GenBank	a,n,nr,k,o,g,au,c,spec	a,n
EMBL	a,n,nr	n,nr
DDBJ	a,n,nr,k	
SWISS-PROT	a,n,nr,d,o,g,au,c	a,n,nr,d,o,g,k,au
PIR	a,nr,k,g,o,au,c,f	n,nr
ENZYME	n,d,spec	n
PDB	a,n,d,k,au,c,spec	n
MMDB	a,n,o,au,c,spec	a,n
PROSITE	a,n,nr,d,au,c,spec	a,n,nr,d,au
PRINTS	n,nr,d,f,s,spec	n
BLOCKS	n,k,s	n,k

Table 3: Search within specific fields in the formbased user interface (UI) and the command-line interface (CL).

a: all text	n: entry name	nr: accession number
d: description	k: keyword	o: organism
g: gene name	au: author	c: citation
f: family	s: sequence	spec: specific fields

Table 4: Explanation of abbreviations in table 3.

browsing the branches of a predefined structure. Most systems support Boolean queries (using and, or, not) as well as wildcards in the text strings. In addition to a form-based query interface, most systems also support command-line querying using the systems query language. A summary is given in table 3. An explanation of the abbreviations used in table 3 is given in table 4. The result of a query is for most systems a list of the data entries that match the query. The actual result data is the complete data entry. In some systems it is possible to define views over the result and in that way one may retrieve only the interesting fields. Two systems, PDB and PIR, supported reuse of query results in new queries.

#### 3 Query language

Our aim is to define a base query language that supports the most common queries that biologists want to ask to biological databanks. According to the interviews and the study of current systems, the most common operations are text search (full-text search and search within specific fields) as well as the application of specialized tools such as sequence alignment tools. Further, we want to be able to enhance the power of the search mechanism by supporting the use of ontologies. As most databanks are accessed via the web we have also investigated web query languages and query languages for semi-structured data such as Lorel (Abiteboul et al. 1007), STRUQL (Fernandez et al. 1997), WebSQL (Mendelzon et al. 1997) and the language defined in (Lambrix and Shahmehri 2000).

As a base for our query language we have defined a simple object model. This is in line with the observation in (Thierry-Mieg et al. 1999) that object models are most convenient when the conceptual model is complex, the data is irregular and the queries are correlated. These characteristics fit biological data well (Elmasri and Navathe 2000). Let C be a set of types and  $\mathcal{R}$  a set of relations. Let  $\mathcal{OID}$  be a set of object identifiers. Then an object o is a tuple  $\langle oid, C, R \rangle$ , where  $oid \in \mathcal{OID}$ ,  $C \in C$  and R is a set of pairs  $\langle r, o' \rangle$  where  $r \in \mathcal{R}$  and o' is an object. Some of the types in C are predefined, such as *String*, *Document* and *Sequence*. For these predefined types a number of operations may be defined.

The base query language we propose is given in an SQL-like syntax. In this section we give examples to show the properties of the language. The syntax and semantics of the complete base language are given in the appendix.

In our language queries can be asked using the type of the requested information. For instance, to find all signal transducers we could write the following query.

#### select \*

#### where Signal-transducer

In this case *Signal-transducer* is a type that is defined in the global view over the data sources or in the ontologies that are used. We require that the query returns all found signal transducers. For instance, assuming that receptors are signal transducers, also the objects belonging to the type *Receptor* should be returned.

The **select**-part of a query can contain \* or a path expression. In the first case all objects satisfying the **where**-part are returned. In the second case we can retrieve selected information. For instance, the following query returns the titles and authors of the articles that are found by following the paths *Reference. Title* and *Reference. Author.*<sup>1</sup> *Reference* and *Title* are relation terms that are defined in the global view or the ontologies. We have also required that the start objects from which the paths should be followed have accession number P18089.<sup>2</sup> We have used the **fills** construct which in the example requires that the value for the *Accession-number* relation is *P18089*.

#### select Reference. Title, Reference. Author where fills Accession-number P18089

It is not always the case that a user knows the internal structure of the databanks. Therefore, path variables are introduced. For instance, the path #p. Title matches any path with Title as the last relation in the path. In the case where we do not know that documents are connected via the *Reference* relation, the previous query can be written as:

#### select #p.Title, #p.Author where fills Accession-number P18089

Conditions in the **where**-part of the query can be combined by boolean operators **and**, **or** and **andnot**. We also allow to require that following some or all paths lead to objects of a specific type. For instance, to find all objects that have a source organism classified as Homo sapiens and have a possible related clone for which the vector type is a phage, we can write:

#### 

Some common queries require operations on special types. To support this, we introduce the **function**-construct. This construct provides a hook to programs implementing operations on predefined types. In the base language we have defined special operations for two kinds of common queries: a string search operation on documents and an alignment operation on sequences. Similar operations could be defined for other data types. As an example, the following query finds all documents containing the string 'E. Coli'.

<sup>&</sup>lt;sup>1</sup>Technically, to be consistent with the object model, the query returns complex objects that have the retrieved values as values for the path expressions. See the semantics in the appendix.

<sup>&</sup>lt;sup>2</sup>The accession number is a commonly used identifier. However, an entity may have different accession numbers in one data source and not all data sources use the same kind of identifier. The accession number used in the query is the SWISS-PROT accession number for the adrenergic alpha-2B receptor.

## select \* where function string-search 'E. Coli'

To find alignments for the sequence 'MDHQD-PYSVQ ...' using BLAST (Altschul et al. 1990) we construct the following query.

# select \* where function align 'MDHQDPYSVQ ...' wrt BLAST

In the case where we leave the decision on which alignment program to use to the retrieval system, the **wrt**-part is left out. All the different types of queries exemplified above can be combined.

The language that we have defined up till now can be extended in a number of ways. For instance, as it was shown in the interviews, it may be that the user has a preference as to which databanks to use to retrieve the information. Similarly, the user may want to use preferred ontologies. The constructs **using-db** and **using-ontology** could be used to provide this flexibility. The retrieval system needs to take these requirements into account during the processing of the query. Another possibility is to allow to prefix relations and types with the names of the databanks and the ontologies.

#### 4 Architecture

Our proposed architecture for a system supporting the query language defined in section 3 is shown in figure 2. The architecture is based on the assumption that different communities create their own databanks (as has been done up till now) and that many legacy systems are in use. Therefore, the integration of the different, possibly heterogeneous, sources such that the multiple sources can be queried in a uniform and integrated way, is required.



#### Figure 2: Architecture

The architecture contains a central system consisting of a user interface, a query interpreter and expander, an answer filter and assembler, and a retrieval engine. Further, the architecture assumes the existence of an ontology base, a databank knowledge base as well as the use of wrappers that encapsulate the source databanks.

#### Ontology base

The ontology base contains a number of ontologies together with information about the dependencies between the items in the different ontologies. The ontologies may be general or de facto standard ontologies (e.g. GeneOntology (GO 2000) that includes a number of the model organisms such as Drosophila melanogaster (fruitfly), Saccharomyces cerevisiae (budding yeast), Mus musculus (mouse), Arabidopsis thaliana and Caenorhabditis elegans), or company-dependent ontologies. The current ontologies use different ways to represent the information. For instance, GeneOntology (GO 2000) has both a flat file and XML format. The TAMBIS ontology (Baker et al. 1999) is based on a description logic. The recently created BioOntology Consortium has proposed DAML+OIL (DAML -, OIL -), a language based on frame systems, description logics and RDF, as their preferred representation language.

The ontologies may be manually created (Lambrix et al. 2002) or semi-automatically using, for instance, dictionary construction methods (e.g. (Soderland et al. 1995, Riloff 1996, Riloff and Jones 1999)) or by merging already existing ontologies (Lambrix and Edberg 2003). The information about dependencies can also be created manually or semi-automatically using, for instance, schema integration methods (e.g. (Mena et al. 2000)).

#### Databank knowledge base

The databank knowledge base stores information about the global integrated view over the data sources as well as data source descriptions and the relations between the global model and the data sources. There are different ways to describe this relation. In the global-as-view approach (e.g. (Roth and Schwarz 1997)) the global concepts are modeled as views on the source models. The advantage is that global queries are easily translated into local queries. However, whenever a local model changes or new sources are added, the global model needs to be updated. In the local-as-view approach (e.g. (Levy et al. 1995)) the local sources are modeled in terms of concepts in the global model. The query planning is harder, but updates in the local models are dealt with in an easier way.

Further, the databank knowledge base stores information about the capabilities of the source databanks. This includes information about query answering capabilities, update frequency and costs for query answering.

Knowledge representation languages with similar requirements have been proposed in the area of information agents such as SIMS (Arens et al. 1993) and the Information Manifold (Levy et al. 1995). The information in the knowledge base could be added manually, but we may also use schema extraction techniques to (semi-)automate this task. The information is used by the retrieval engine to decide which databanks to use.

#### Central system

The user interacts with the central system via the user interface. User queries are stated via the interface, translated into queries in the language as proposed in section 3, and sent to the query interpreter and expander. The query can be expanded to take domain knowledge into account. This domain knowledge could be found in the ontologies. For instance, if the domain knowledge states that receptors are signal transducers and the system needs to retrieve all signal transducers with particular characteristics, then the query may be expanded to also retrieve the receptors with these characteristics. The used domain knowledge may be of different kinds such as strict subsumption knowledge and default knowledge.

The expanded query is sent to the retrieval engine. The retrieval engine generates the answer to the query. It processes the expanded query by generating a query plan. The query plan consists of subqueries to the (wrappers of) the different databanks. For generating the query plan the engine needs to decide which databanks should be used and in which order. This decision is based on information about the contents, the availability and the query capabilities of the different databanks as well as domain knowledge. For instance, information may be obtained from different sources or some sources may not be available at query time. This information can be obtained from the databank knowledge base as well as from the ontology base. Further, the query plan should be optimized with respect to the query. According to (Davidson et al. 1995) a number of strategies should be used including pushing operations down to the source databanks as much as possible and exploiting parallelism at the servers.

The answer filter and assembler receives the results of the queries from the retrieval engine and may modify the results and arrange them for presentation to the user.

#### Source databanks

The source databanks are accessed through the wrappers. A wrapper encapsulates a databank and deals with the translation of the sub-queries from the central system into queries that are supported by the databank's retrieval system. The databank's retrieval system can, for instance, be a relational database management system or a collection of PERL programs. The results of the local query are then propagated to the retrieval engine of the central system.

#### 5 Related work

There exist a number of systems that provide access to multiple biological databanks. These systems can be divided into two categories (Davidson et al. 2001). The category of the link-driven federations contains most of the currently used websites that provide an interface to multiple biological resources, such as Entrez (Entrez -) and SRS (SRS -). These systems support a number of basic queries via a web interface. Often, they also allow to use alignment algorithms such as BLAST (Altschul et al. 1990) on the result of a query. Usually, the users need to explicitly state which resources should be used for retrieving the answers, requiring good knowledge of the underlying sources. The data source systems are often implemented using flat files and specialized retrieval packages. Most of the integration is link driven and is achieved by the creation of cross-reference indexes. For instance, the SRS language (SRSum -) defines search in indexes of databanks (including string search, regular expressions, numeric ranges and dates), and combinations of queries using and, or and andnot. With respect to the combination of databanks the *link* construct is introduced. This allows for queries of the forms 'find all entries in databank A that are referenced in databank B', and 'find all entries in databank A that reference entries in databank B'. The advantage of such systems is that queries relating to knowledge in different databanks can be asked and that the query processing is fast. However, although this is a first step in integrating data sources, this solution does not handle the differences in terminology used in the underlying sources, is syntax based and only allows limited query functionality over multiple databanks. Also, adding a new resource requires cross-referencing with the other resources.

The BioKleisli (Davidson et al. 1997), K2 (Davidson et al. 2001), TINet (Eckman et al. 2001), P/FDM (Kemp et al. 2000) and TAMBIS (Goble et al. 2001) systems, our own approach and the proposal in (Burger et al. 1997) use an approach based on view integration. Also IBM's DiscoveryLink (DiscoveryLink -) can be placed into this category. As mentioned before, in this case the underlying schemas

are integrated to form a global schema. The global schema is queried in a high-level language such as CPL (BioKleisli) or OQL (K2). For instance, OQL (Catell et al. 2000) relies on the ODMG object model. It is close to SQL92 and has extensions concerning complex objects, object identity, path expressions, polymorphism, operation invocation and late binding. One difference between our query language and the other proposals is that our language has been inspired by web query languages and is based on a (currently small) study of the use of current biolog-ical databanks. Also the TAMBIS developers did a user requirements survey (Stevens et al. 2001). The other proposals seem to be based mainly on database technology. In general, the view approach allows for more complex querying and allows for support for integration on schema level. The advantages of such systems include the possibility of complex querying, the knowledge that is required of the end-user is not as large and the local conceptual models are used in the integration. Further, these view integration systems may also be used to create warehouses. An extension to the current implemented systems that is needed and that we propose, is the use of ontologies. This would lead to a possible solution of the discrepancy problem in the local schemas as well as it is a step towards semantic querying. A system that has proposed some solutions in that direction is TAMBIS (Goble et al. 2001). Some of the proposals in this category are implemented and some of them have been field tested. We are currently implementing a prototype based on our proposed architecture.

#### 6 Conclusion

In this paper we introduced, based on a study of user needs and current practise, a base query language with operations that should be available in query languages for multiple biological databanks. The query language provides an integrated and centralized access that allows for complex queries in a highly distributed and heterogeneous environment. Further, we introduced a possible architecture for systems supporting this integrated access.

An important direction for future work is the definition of a knowledge representation language and an inference mechanism that support the representation of and reasoning about the information about the contents and capabilities of the databanks as well as the information in the ontology base. Another important direction for future work is the definition, maintenance and integration of ontologies. The use of ontologies can greatly improve the quality of search results. An issue that we did not tackle in this paper is the updating and maintenance of the databanks. In the integrated system this will become an important aspect.

#### Acknowledgements

We acknowledge the financial support of CENIIT (Center for Industrial Information Technology).

#### Appendix

The syntax of the language is as follows.<sup>3</sup>

<query> :: select <selection-list> [where <where-list>]

 $<sup>^{3}\{\}</sup>$  represents an occurrence of zero or more times. [] represents an optional part.

<selection-list> ::

 $| < path > \{, < path > \}$ 

 $\langle \text{where-list} \rangle ::$ 

<type-description> <function-description> (<where-list> and <where-list> {and <where-list>}) | (<where-list> or <where-list> {or <where-list>}) (<where-list> and-not <where-list>)

<function-description> :: <content-description> | <alignment-description>

<type-description> ::  $\operatorname{top}$ <type-name> all <path> (<query>) some <path> (<query>) atleast <non-negative integer> <path> atmost <non-negative integer> <path> fills <path> <object-name> {<object-name>}

<content-description>:: function string-search '<string>'

<alignment-description>:: function align <sequence-name> wrt <program-name>

<path> :: <path-part> {. <path-part>} <path-part> :: #<string> | <relation-name> <relation-name> :: <string> <type-name> :: <string> <object-name> :: <string> <sequence-name> :: <string> <program-name> :: <string> <string> :: <symbol>[<string>] <symbol> :: A | B | .. | Z | a | b | .. | z | 0 | .. | 9

The semantics of the query language is defined in a model-theoretic way. An interpretation for the language consists of a tuple  $\langle \mathcal{D}, \varepsilon \rangle$ , where  $\mathcal{D}$  is a domain and  $\varepsilon$  an extension function. The extension function maps relations into sub-sets of  $\mathcal{D} \times \mathcal{D}$ , types into sub-sets of  $\mathcal{D}$  and objects into elements of  $\mathcal{D}$  such that  $\varepsilon[o_1] \neq \varepsilon[o_2]$  whenever  $o_1 \neq o_2$ . We require that if  $o = \langle oid, C, R \rangle$ , then  $\varepsilon[o] \in \mathbb{C}$ 

 $\varepsilon[C]$ . If o contains  $\langle r, o' \rangle$  in its R-component then we require that  $\langle \varepsilon[o], \varepsilon[o'] \rangle \in \varepsilon[r]$ . For path expressions, the following holds:

 $\begin{array}{l} \varepsilon[r_1.r_2.\ldots,r_n] = \{ < x,y > \mid \exists \ x_1,\ldots,x_{n-1}: \\ < x,x_1 > \in \varepsilon[r_1] \land < x_1,x_2 > \in \varepsilon[r_2] \land \ldots \\ \land < x_{n-1},y > \in \varepsilon[r_n] \}. \\ \text{Also, } \varepsilon[\#\mathbf{q}] = \{ < x,y > \mid \exists \ p: \ < x,y > \in \varepsilon[p] \} \end{array}$ 

#### query descriptions:

 $\varepsilon$ [select \* where A] =  $\varepsilon$ [A]

 $\varepsilon$ [select p where A] =  $\{ \begin{matrix} x \mid \exists \ y \in \varepsilon[A]: \ < y, \ x > \in \varepsilon[p] \ \}$ 

 $\begin{array}{l} \varepsilon[\textbf{select} \ p_1, ..., p_n \ \textbf{where} \ A] = \\ \{\pi_{p_1, ..., p_n}(x) \ | \ x \in \varepsilon[A]\} \ \text{for} \ n > 1 \end{array}$ 

i)  $\pi_{p_1,\ldots,p_n}(x)$  is the projection of x on the paths  $\begin{array}{l} p_1, \dots, p_n, \\ \text{ii}) \forall p \in \{p_1, \dots, p_n\} \forall z; \\ < x, z > \in \varepsilon[p] \leftrightarrow < \pi_{p_1, \dots, p_n}(x), z > \in \varepsilon[p] \end{array}$ 

iii)  $\forall p \notin \{p_1, \dots, p_n\} \forall z: \langle \pi_{p_1, \dots, p_n}(x), z \rangle \notin \varepsilon[p]$  $\varepsilon[A \text{ and } B] = \varepsilon[A] \cap \varepsilon[B]$  $\varepsilon[A \text{ or } B] = \varepsilon[A] \cup \varepsilon[B]$  $\varepsilon[A \text{ and-not } B] = \varepsilon[A] \setminus \varepsilon[B]$ 

#### type descriptions:

 $\varepsilon[top] = \mathcal{D}$ 

 $\begin{array}{l} \varepsilon[\mathbf{all}\ p\ A] = \\ \{x \mid \forall\ y: <\!\! x,\!\! y\!\! > \in \varepsilon[p] \rightarrow y \in \varepsilon[A]\} \end{array}$ 

 $\varepsilon$ [some p A] =  $\{x \mid \exists y: \langle x, y \rangle \in \varepsilon[p] \land y \in \varepsilon[A]\}$ 

 $\begin{array}{l} \varepsilon[ \textbf{atleast} \ m \ p] = \\ \{x \mid \sharp \ \{y \mid <\!\! x,\!\! y\!\!> \in \varepsilon[p]\} \geq m\} \end{array}$ 

$$\begin{array}{l} \varepsilon[ \textbf{atmost } m \ p ] = \\ \{x \mid \sharp \ \{y \mid <\!\! x,\!\! y\!\! > \in \varepsilon[p]\} \le m\} \end{array}$$

 $\begin{array}{l} \varepsilon[\mathbf{fills}\ p\ o_1\ \dots\ o_m] = \\ \{x\ |\ <\!\!x, \varepsilon[o_1]\!> \in \varepsilon[p] \land \ \dots\ \land <\!\!x, \varepsilon[o_m]\!\!> \in \varepsilon[p]\} \end{array}$ 

#### content description:

 $\varepsilon$ [function string-search s] =  $\{x \mid x \in \varepsilon [Document] \land x \text{ contains } s\}$ 

#### alignment description:

 $\varepsilon$ [function align s wrt pr] =  $\{ x \mid x \in \varepsilon[Sequence] \land sim(x,s,pr) \}$ sim(x,s,pr) is true if x and s are similar with respect to the similarity defined by pr.

 $\varepsilon$ [function align s] =  $\{x \mid x \in \varepsilon[Sequence] \land \exists pr: sim(x,s,pr)\}$ 

#### References

- Abiteboul, S., Quass, D., McHugh, J., Widom, J., and Wiener, J., The Lorel query language for semistructured data, International Journal on Digital Libraries, 1(1):68-88, 1997.
- Altschul, S.F., Gish, W., Miller, W., Myers, E.W., and Lipman, D.J., Basic Local Alignment Search Tool, Journal of Molecular Biology, 215:403-410, 1990.
- Arens, Y., Chee, C.Y., Hsu, C., and Knoblock, C., Retrieving and integrating data from multiple information sources, International Journal on Intelligent and Cooperative Information Systems, 2(2):127-158, 1993.
- Attwood, T., The PRINTS database: a resource for identification of protein families, Briefings in Bioinformatics, 3(3):252-263, 2002.http://www.bioinf.man.ac.uk/dbbrowser/PRINTS/
- Bairoch, A., The ENZYME database in 2000, Nucleic Acids Research, 28:304-305, 2000http://www.expasy.ch/enzyme/
- Bairoch, A., and Apweiler, R., The SWISS-PROT protein sequence database and its supplement in TrEMBL in 2000, Nucleic Acids Research, 28:45-48, 2000. http://www.expasy.ch/sprot/

- 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.
- Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp B.A., and Wheeler, D.L., Genbank, *Nucleic Acids Research*, 28(1):15-18, 2000. Accessible via Entrez (Entrez -).
- Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shindyalov, I., and Bourne, Ph., The Protein Data Bank, *Nucleic Acids Research*, 29(1), 235-242, 2000. http://www.rcsb.org/pdb/
- Burger, E., Link, J., and Ritter, O., A Multi-Agent Architecture for the Integration of Genomic Information, First International Workshop on Intelligent Information Integration (in conjunction with KI97), 1997.
- Catell, R., Barry, D., Berler, M., Eastman, J., Jordan, D., Russel, C., Schadow, O., Stanienda, T., and Velez, F., *The Object-Oriented Standard ODMG* 3.0, Morgan Kaufmann Publishers, 2000.
- DAML, DARPA Agent Markup Language, http://DAML.SemanticWeb.org/
- Davidson, S., Crabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., and Stoeckert, C., K2/Kleisli and GUS: Experiments in integrated access to genomic data sources, *IBM Systems Journal, Issue on Deep computing for the life sciences*, 40(2):512-531, 2001.
- Davidson, S., Overton, C., and Buneman, P., Challenges in Integrating Biological Data Sources, Journal of Computational Biology, 2:557-572, 1995.
- Davidson, S., Overton, C., Tannen, V., and Wong, L., BioKleisli: A digital library for biomedical researchers. *Journal of Digital Libraries*, 1(1):36-53, 1997.
- DBCAT, The Public Catalog of Databases, http://www.infobiogen.fr/services/dbcat
- DDBJ, DNA DataBank of Japan, http://www.ddbj.nig.ac.jp/
- DiscoveryLink, http://www-3.ibm.com/ solutions/lifesciences/solutions/discoverylink.html
- Eckman, B., Kosky, A., Laroco, L., Extending traditional query-based integration approaches for functional characterization of post-genomic data, *Bioinformatics*, 17(7):579-670, 2001.
- Elmasri, R. and Navathe, S., *Fundamentals of database systems*, 3rd. ed., ch 27.5 Genome Data Management, Addison-Wesley, 2000.
- Entrez, http://www.ncbi.nlm.nih.gov/Entrez/
- Falquet L., Pagni M., Bucher P., Hulo N., Sigrist, C., Hofmann K., and Bairoch, A., The PROSITE database, its status in 2002, *Nucleic Acids Research*, 30:235-238, 2002. http://www.expasy.ch/prosite/
- Fernandez, M., Florescu, D., Levy, A., and Suciu, D., A Query Language for a Web-Site Management System, ACM SIGMOD Record, 26(3):4-11, 1997.

GeneLynx, http://www.genelynx.org/

- The GeneOntology Consortium, GeneOntology: tool for the unification of biology, *Nature Genetics*, 25:25-29, 2000. http://www.geneontology.org/
- Goble, C., Stevens, R., Ng, G., Bechhofer, S., Paton, N., Baker, P., Peim, M., and Brass, A., Transparent access to multiple bioinformatics information sources, *IBM Systems Journal, Issue on Deep computing for the life sciences*, 40(2):532-551, 2001.
- Henikoff, J., Greene, E., Pietrokovski, S., and Henikoff, S., Increased coverage of protein families with the BLOCKS database servers, *Nucleic Acids Research*, 28(1):228-230, 2000 http://BLOCKS.fhcrc.org/
- KEGG, Kyoto Encyclopedia of Genes and Genomes, http://star.scl.genome.ad.jp/kegg/
- Kemp, G., Angelopoulos, N., Gray, P., A schemabased approach to building a bioinformatics database federation, *Proceedings of the IEEE In*ternational Symposium on Bioinformatics and Biomedical Engineering, pp 13-20, 2000.
- Lambrix, P., and Edberg, A., Evaluation of ontology merging tools in bioinformatics, *Proceed*ings of the Pacific Symposium on Biocomputing - PSB03, Kauai, Hawaii, USA, 2003.
- Lambrix, P., Habbouche, M., and Pérez, M., Evaluation of ontology engineering tools for bioinformatics, forthcoming, 2002.
- Lambrix, P., and Shahmehri, N., Querying Documents using Content, Structure and Properties, Journal of Intelligent Information Systems, 15(3):287-307, 2000.
- Levy, A., Srivastava, D., and Kirk, T., Data Model and Query Evaluation in Global Information Systems, Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval, 5(2):121-143, 1995.
- Mena, E., Illamarendi, A., Kashyap, V., and Sheth, A., OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies, *Journal* of Distributed and Parallel Databases, 8(2):223-271, 2000.
- Mendelzon, A., Mihaila, G., and Milo, T., Querying the World Wide Web International Journal on Digital Libraries, 1(1):54-67, 1997.
- NAR, Nucleic Acids Research, 29(1), 2001.
- OIL, Ontology Interface Layer, http://www.ontoknowledge.org/oil/
- Riloff, E., An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains, AI Journal, Vol 85(1-2), pp 101-134, 1996.
- Riloff, E. and Jones, R., Learning dictionaries for information extraction by multi-level bootstrapping, *Proceedings of the National Conference* on Artificial Intelligence, AAAI-99, pp 474-479, 1999.
- Roth, M., Schwarz, P., Don't scrap it, wrap it! A wrapper architecture for legacy data sources, *Proceedings of the Twenty-third International Conference on Very Large Databases*, 266-275, 1997.

Soderland, S., Fisher, D., Aseltime, J., and Lehnert, W., CRYSTAL: Inducing a Conceptual Dictionary, Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp 1314-1319, 1995.

SRS, http://srs.ebi.ac.uk/

SRS Users Manual version 6.07.

- Stevens, R., Goble, C., Baker, P., and Brass, A., A classification of tasks in bioinformatics, *Bioinformatics*, 17(2):180-188, 2001.
- Stoesser, G., Baker, W., van den Broek, A., Camon, E., Garcia-Pastor, M., Kanz, C., Kulikova, T., Lombard, V., Lopez, R., Parkinson, H., Redaschi, N., Sterk, P., Stoehr, P., and Tuli, M., The EMBL Nucleotide Sequence Database, *Nucleic Acids Research*, 29(1):17-21, 2001. http://www.ebi.ac.uk/embl/.
- Thierry-Mieg, J., Thierry-Mieg, D., and Stein, L., ACEDB: The ACE database manager, chapter 22 in Letovsky, S., ed., *Bioinformatics* -*Databases and Systems*, Kluwer Academic Publishers, 1999.
- Wang, Y., Anderson, J., Chen, J., Geer, L., He, S., Hurwitz, D., Liebert, C., Madej, T., Marchler, G., Marchler-Bauer, A., Panchenko, A., Schoemaker, B., Song, J., Thiessen, P., Yamashita, R., and Bryant, S., MMDB: Entrez's 3D-structure database, *Nucleic Acids Research*, 30(1):249-252, 2002. Accessible via Entrez (Entrez -).
- Wu, C., Huang, H., Arminski, L., Castro-Alvear, J., Chen, X., Hu, Z.-Z., Ledley, R., Lewis, K., Mewes, H.-W., Orcutt, B., Suzek, B., Tsugita, A., Vinayaka, C., Yeh, L.-S., Zhang, J., and Barker, W., The Protein Information Resource: an integrated public resource of functional annotation of proteins, *Nucleic Acids Research*, 30(1):35-37, 2002. http://pir.georgetown.edu/