A Data Model and Query Language to Explore Enhanced Links and Paths in Life Science Sources

George Mihaila
IBM T.J. Watson Research Center
mihaila@us.ibm.com

Felix Naumann
Humboldt-Universität zu Berlin
naumann@informatik.hu-berlin.de

Louiqa Raschid
University of Maryland
louiqa@umiacs.umd.edu

Maria Esther Vidal
Universidad Simón Bolívar
mvidal@ldc.usb.ve

ABSTRACT

Links in life science sources capture important domain knowledge. However, current simple physical link implementations are not rich in either representation or semantics. This paper proposes the *e-link* framework and tools to assist scientists in exploring and exploiting the knowledge that should be captured in links.

1. INTRODUCTION

An abundance of Web-accessible bio-molecular data sources contain data about scientific entities, such as genes, sequences, proteins and citations. The sources have varying degrees of overlap in their content and they are richly interconnected to each other. Experiment protocols to retrieve relevant data objects (data integration queries) explore multiple sources and traverse the links and the paths (informally concatenations of links) through these sources. While such navigational queries are critical to scientific exploration, they also pose significant limitations and challenges.

The key limitation is that current physical link implementations are inherently poor with respect to both syntactic representation and semantic knowledge. We illustrate using an example. OMIM is a source that has knowledge on human genes and genetic disorders. Each entry in OMIM may have links to entries in multiple other sources. While there is significant knowledge and curation effort associated with the creation of each of these links, this knowledge is not explicitly captured in the link. All links appear to occur at the level of the OMIM entry. In a later section, we discuss many examples of specific sub-elements within the OMIM entry that are actually associated with the link; this is additional knowledge that is useful to the scientist. Similarly, suppose we consider two or more links from OMIM entries to say proteins in SWISSPROT. These links do not explicitly specify the underlying relationship that led to the creation of this link and one may assume that the links capture the same relationship. However, a scientist who examines the OMIM and SWISSPROT entries may conclude that the relationships that have been captured are quite different.

Copyright is held by the author/owner. Eighth International Workshop on the Web and Databases (WebDB 2005), June 16-17, 2005, Baltimore, Maryland.

Links between entries in the sources are created for many different reasons. Biologists capture new discoveries of an experiment or study using links, whereas data curators add links to augment, to complete or to make consistent the knowledge captured among multiple sources. For example, a result reported in a paper in PUBMED may lead a curator to insert a link from a data entry in say OMIM to this citation in PUBMED. Algorithms insert links automatically when discovering similarities among two data items, e.g., to represent sequence similarity following a BLAST search. Thus, the simple unlabeled physical links that are in use today are insufficient to represent subtle and diverse relationships.

In this research, we propose the *e-link* framework and methodology and tools to assist scientists in exploring and exploiting the knowledge that should be captured in links. To do so we must accomplish the following three objectives. In this paper, we address the first two objectives.

- Develop a data model that can represent sources, data objects and the enhanced semantic links (*e-links*) between data objects.
- Develop a query language and query evaluation engine for scientists to meaningfully explore these semantically enhanced *e-links* and paths.
- Develop (machine-assisted) techniques to extract, generate and label existing links to create *e-links*.

We briefly review related research. Clearly there is much related work in knowledge representation. RDF and XML Topic Maps provide a rich conceptual framework and expressive query languages that can be applied to represent the *e-link* framework. Ontologies also provide a framework to capture the semantics of links. Our objective in this paper is to focus on the simple framework needed for *e-links* and we expect to freely exploit ideas from the richer frameworks.

The three major repositories, NCBI, DDBJ and EBI have recently made significant efforts to provide integrated access to the many entries and links between entries that exist in the sources that they manage. Examples include ELink [1], RefSeq and LocusLink at NCBI; LinkDB [2], and Integr8 [3].

These projects focus on providing unified access to the links but do not attempt to enhance the representation and the semantics of links.

There are other projects that target and enhance specific links. For example, the PDBSProtEC project [4, 12] is a resource to link PDB chains with SwissProt codes and EC numbers. We expect that there will be many such efforts to enhance specific links. Our *elink* framework is generic and can in principle be applied to any enhanced link; we would of course need to develop the machinery to interpret such enhanced links and translate to our *e-link* framework.

Recent work in [7, 11] present sophisticated query languages to explore knowledge in interconnected data sources, beyond simple navigational queries. For example, the IBM DB2 Graph Extender supports complex queries on large object graphs and can discover associations important to systems biology, e.g., across genome comparisons. These projects can be extended to accommodate the enhanced semantics of the *e-link* framework.

The paper is organized as follows: Section 2, describes a simple model for life science sources and presents examples of enhanced links. In Sec. 3, we present the data model for the *e-link* framework and in Sec. 4, we (informally) present the query language. Sec. 5 considers the semantics of queries and Sec. 6 describes the steps of query evaluation.

2. MODELING LIFE SCIENCES SOURCES

We first describe a simple model for life science sources. The model was first presented in [8, 9] where we investigated interesting metrics to characterize life science sources.

2.1 A Simple Model for Life Science Sources

Life science sources may be modeled at three levels: the physical level, the object level and the ontology 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 Fig. 1.

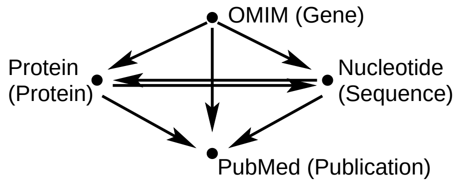

Figure 1: A Source Graph for NCBI Data Sources (and Corresponding Scientific Entities)

The sources are a subset of sources at the National Center for Biotechnology Information (NCBI) and can be accessed at `http://www.ncbi.nlm.nih.gov`. The sources are PubMed, Protein, Nucleotide, and Omim (not an NCBI source). The physical level is modeled by a directed *Source Graph*, where nodes represent data sources and edges represent a physical 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 associated with a disease in Omim links to multiple citations in PubMed. An *Object Graph* as shown in Fig. 2 represents the data objects of the sources and the object links between the objects. Thus, each link in the *Source Graph* corresponds to a collection of object links of the *Object Graph*, each going from a data object in one source to another object, in the same or a different source. Note that links in *Source Graph* can be bi-directional (though not always symmetric) and *Source Graph* may be cyclic.

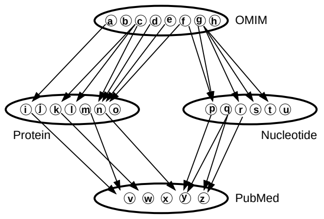

Figure 2: An Object Graph for NCBI Data Sources with Data Entries (Objects) and Links

The ontology 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 of *Source Graph*. For example, the class *Citation* may be implemented by the data source PubMed. A source of *Source Graph* typically provides a unique identifier for each of the entities or objects in *Object Graph* and includes attribute values that characterize them. Table 1 provides a mapping from the logical classes to some physical data sources of some *Source Graph*.

CLASS	DATA SOURCE
Sequence (s)	NCBI Nucleotide database EMBL Nucleotide Sequence database DDBJ
Protein (p)	NCBI Protein database UniProt SwissProt
Citation (c)	NCBI PubMed NCBI Book

Table 1: A Possible Mapping from Ontology Classes to Physical Data Sources of *Source Graph*

2.2 Enhancing Links Among Data Entries

We present examples to illustrate that the the simple unlabeled physical links that are in use today are insufficient to represent diverse relationships.

Consider a SwissProt entry with a link to an Omim entry; it is illustrated in Fig. 3. In the flat structure of the SwissProt entry, this link is represented by embedding an Omim ID as a top-level attribute of the entry, and the entry may include an HTML hyperlink to the Omim entry. Such a link neither represents the sub-element of the SwissProt entry to which the link refers, nor the sub-element of the Omim entry to which the link points, nor does it represent the reason to insert this link. Biologists examining the SwissProt entry rely on their experience and can infer these link properties after a time-consuming examination. Machines and algorithms cannot perform such analysis at the necessary level of detail and precision. In this particular case, the *e-link* should not originate from the SwissProt entry; instead the "real" origin is the CC-DISEASE attribute within that entry. The *e-link* should also not represent a generic relationship; it should be labeled as a *causal* relationship, telling humans and machines that *the protein in question is known to cause the disease* pointed to by the *e-link*.

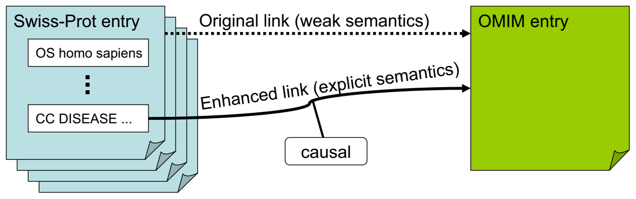
Swiss-Prot entry
OS homo sapiens
CC DISEASE ...
Original link (weak semantics)
Enhanced link (explicit semantics)
causal
OMIM entry

Figure 3: The Enhanced *e-link* from Swiss-Prot to OMIM

We note that determining the semantics and labels of *e-link*s for some physical link between two sources may not be straightforward, and scientists may not always reach a consensus as to the desired semantics. Nevertheless, we believe that the significant activity related to ontologies for the life sciences, and the resulting advances in establishing controlled vocabularies to describe functionality and relationships among concepts, e.g., the GO Ontology [5] and GOA [6] will contribute towards the success of our research.

Consider the physical link from the origin source UniProt to the target source Omim illustrated in Fig. 4. The physical link instances between UniProt and Omim entries corresponds to two distinct *e-links* with different semantics. Both *e-links* originate in the same sub-element of UniProt. One *e-link* has the meaning *is causal for disease* and the target sub-element in Omim is CLINICALFEATURES. The second *e-link* has the meaning *describes genetic defects* and the target sub-element in Omim is MAPPING. In this example, the original physical link of the *Source Graph* is classified as two *e-links*, whose target sub-element in Omim is different, and where the two *e-links* have different meaning.

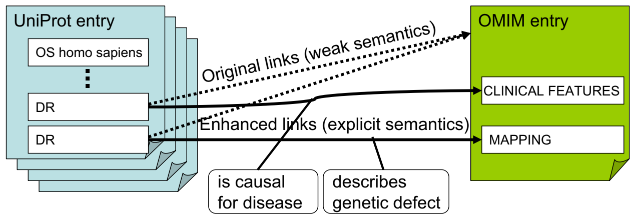

Figure 4: Enhancing a Link from UniProt to Omim to Produce Two *e-links* with Different Target Sub-Elements in Omim

Next, consider the link from the origin source UniProt to the target source GO in Fig. 5. This physical link captures three *e-links*, where the origin sub-element and the meaning of the three *e-links* is different. The target is the GO entry. The first *e-link* has meaning *has (sub)cellular location* and the origin sub-element in UniProt is SUBCELLULAR. The second *e-link* has the meaning *has molecular function* and the origin sub-element in UniProt is MOLFUNC and the third *e-link* has meaning *participates in biological process* and the origin sub-element in UniProt is BIOLPROC.

Finally, we consider the case where different physical links between different data sources appear to have the same meaning. There are six physical links in the *Source Graph*, each originating in the same sub-element of UniProt. The target of each link is a data entry in one of six different protein data sources, InterPro, Pfam, SMART, PROSITE, PRINTS, and TIGRFAMS; the links are illustrated in Fig. 6.

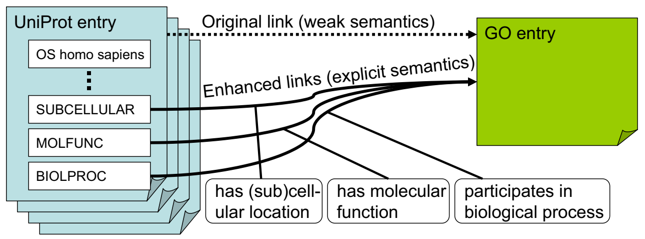
UniProt entry
OS homo sapiens
SUBCELLULAR
MOLFUNC
BIOLPROC
Original link (weak semantics)
Enhanced links (explicit semantics)
GO entry
has (sub)cellular location
has molecular function
participates in biological process

Figure 5: Enhancing a Link from UniProt to GO to Produce Three *e-links* with Different Origin Sub-Elements in UniProt

While the physical links are between different sources, they each have the meaning *contains a sequence signature*. This example of six physical links producing potentially six *e-links*, all of which are equivalent with respect to meaning is a frequent occurrence in life science sources, because the contents of sources overlap and the sources are richly interconnected. This motivates our research on a data model that can specify the equivalence of *e-link*s that are of the same link type.

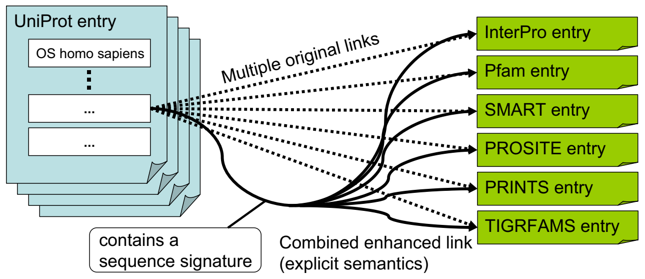

Figure 6: Enhancing a Link from UniProt to Six Protein Data Sources

3. THE DATA MODEL

In this section we formalize the life sciences graphs introduced in the previous section. The data model we consider is comprised of three graphs, which capture the various abstraction levels: the *Ontology Graph* at the logical level and the *Source Graph* and *Object Graph* at the physical level. This is an extension of our previous work on modeling life science sources [8, 9].

An *Ontology Graph* is a graph (C, LT), where C is a set of logical classes (e.g., protein, gene, citation) and LT is a set of link types between logical classes. A link type is a triple (C_1, l, C_2) where C_1 is the origin class, C_2 is the target class and l is a label from a set of link labels L. For example (citation, *describesBehaviorOf*, gene) is a link type between citations and genes. We note that at this stage we only consider a simple label to capture the semantics of a link; in future work we expect to consider the use of ontologies with their richer semantics.

A *Source Graph* is a graph (S, L_S), where S is a set of sources which store instances of logical classes and L_S is a set of source links which implement link types. Each source s stores instances of a single logical class, denoted $m_S(s)$. A source link in LS is a triple (s_1, l, s_2) such that there exists a link type $(m_S(s_1), l, m_S(s_2))$ in LT. Given a typed link (e.g., *describesBehaviorOf*) between class C_1 (e.g., citation) and class C_2 (e.g., gene), a pair of sources (S_1, S_2) where S_1 (PubMed) contains objects of the logical class C_1 and S_2 (UniGene) contains objects of target logical class C_2, can implement this typed link. Note that the set of implemented link types, e.g., (PubMed *describesBehaviorOf*, UniGene) is made public by the sources. Thus, PubMed will typically advertise that it has links of this type. If the reverse links are stored by UniGene, it too will advertise the link. We note that the links between the objects may not be symmetric.

An *Object Graph* is a graph (O, L_O), where O is a set of objects and L_O is a set of links between objects. There is also a mapping $m_O : O \rightarrow S$ defining for each object o, the source $m_O(o)$ where o is physically stored. A given object of class C_1 in source S_1 can have a link with a label l (e.g., describesBehaviorOf) to another object of class C_2 in source S_2 only if there is a source link (S_1, l, S_2) in L_S.

In addition we have a link concatenation matrix LL which specifies the meaningful concatenations of link types: if (lt_1, lt_2) is a pair in LL, then the target class of lt_1 is the same as the origin class of lt_2 and the concatenation of lt_1 with lt_2 is meaningful. We note that for now we only consider pairs of links.

We are now ready to integrate the ontology, source and object graphs into a single, unified data model.

DEFINITION 1. *The* e-link *data model is a 9-tuple* $= (C, L, LT, S, L_S, O, L_O, m_S, m_O, LL)$ *where:*

- C *is a set of classes*
- L *is a set of link labels;*
- LT *is a set of link types;*
- S *is a set of sources*
- L_S *is a set of links between sources;*
- O *is a set of objects;*
- L_O *is a set of links between objects;*
- m_S *is a mapping from* S *to* C*;*
- m_O *is a mapping from* O *to* S*;*
- LL *is a set of pairs of link types.*

4. THE QUERY LANGUAGE

We define the *e-link* query language as a regular expression over the alphabet $C \cup L$ where each class occurrence can optionally be annotated with a predicate expression. The BNF specification of the language syntax is given in Fig. 7.

The result to all queries are paths in the *Source Graph* or the *Object Graph*, where a node in the graph is also a path. Some of the uses of the query language are as follows:

- Identify sources in the *Source Graph* that implement a given class: For example, to find sources that implement class "publication ", one can submit the query $Q_1 =$ `publication`.

- Identify sources in the *Source Graph* that implement a given link type: An example is a source that contains *proteins that are linked to an entry in the RefSeq database.* Assuming a link type (`protein`, *linkedToEntryInRefSeq*, `refseqentry`) in LT. The query $Q_2 =$ `protein` *linkedToEntryInRefSeq* retrieves all sources that contain protein entries that are linked to an entry in the RefSeq database.

- Identify paths in the *Source Graph* using wildcards: The symbols ϵ_C and ϵ_L are wildcards matching any class and any link type respectively. For example, to retrieve all the sources linked to the PubMed source by any link type, one can write a query such as the following:
  $Q_3 =$`publication`[*source* = "PubMed"] ϵ_L ϵ_C;

Query	:=	"(" Query ")"
	|	Query "|" Query
	|	Query Query
	|	ϵ_C
	|	ϵ_L
	|	Term
Term	:=	ClassName Annotation
	|	LinkLabel
Annotation	:=	empty
	|	"[" Condition "]"
Condition	:=	"(" Condition ")"
	|	Condition "and" Condition
	|	Condition "or" Condition
	|	"not" Condition
	|	Field
	|	Field Op Value
Op	:=	"=" | "$\neq$" | ">" | "<"
	|	"$\geq$" | "$\leq$" | "contains"

Figure 7: The Syntax of the Query Language

- Identify paths in the *Source Graph* that satisfy a path regular expression: For example the query $Q_4 =$ `publication` (*describes* | *describes describes*) retrieves all sources that contain classes connected by one or two *describes* links starting from sources containing publications.

- Identify paths in the *Object Graph* that satisfy a path regular expression that can include source/object predicates: For example the query $Q_5 =$ `publication`[*author* = "John Smith" and *title* contains "cancer"] *describes* `protein`[*source* = "RefSeq"] would retrieve publications written by Smith whose title contains the term "cancer" and which describe RefSeq proteins.

5. QUERY LANGUAGE EVALUATION

Intuitively, a query can be evaluated in five steps.

Step 1. Enumerate simple path expressions matching the regular path expression Q (the predicate expressions are carried over). For example, the query $Q =$ `publication` (*describes* | *describes describes*) is expanded as follows:

`publication` *describes*
`publication` *describes describes*

Step 2. For each such simple path expression e, insert ϵ_L in between each pair of consecutive class labels and ϵ_C in between each pair of consecutive link labels to obtain e'. If the simple path expression ends with a link label, append the ϵ_C symbol at the end. This yields the following:

`publication` *describes* ϵ_C
`publication` *describes* ϵ_C *describes* ϵ_C

Step 3. Typecheck each e': verify if each triple of consecutive symbols of the form $c_1 l_1 c2$ corresponds to an actual link type $(c_1, l_1, c_2) \in LT$ and if each triple of consecutive symbols of the form $l_1 c_1 l_2$ corresponds to a pair $(l_1, l_2) \in LL$. Also, if there are wildcards, replace ϵ_C by all possible class and ϵ_L by all possible link types that result in valid path expressions (according to the typechecking rules above).

Suppose that LT specifies that `publication` can only describe `publication`, `protein` or `gene`, and LL includes (*describes*, *describes*). This step yields the following:

`publication` *describes* `publication`
`publication` *describes* `protein`
`publication` *describes* `gene`
`publication` *describes* `publication` *describes* `publication`
`publication` *describes* `publication` *describes* `protein`
`publication` *describes* `publication` *describes* `gene`

Step 4. For each valid e' find all the actual *source path* instances matching e' in the *Source Graph* that also satisfy all the source predicates. A source path instance in the *Source Graph* is said to match a simple path expression if there exists a mapping from the set of all the sources in this path instance to the class labels in the path expression. For a particular *Source Graph*, this step could yield:

PubMed *describes* PubMed
PubMed *describes* SwissProt
PubMed *describes* UniGene
PubMed *describes* PubMed *describes* PubMed
PubMed *describes* PubMed *describes* SwissProt
PubMed *describes* PubMed *describes* UniGene

Step 5. For each source path, evaluate a query evaluation plan (for this path) against the *Object Graph*.

6. QUERY EVALUATION

We provide a sketch of the steps for query evaluation. We then provide a brief description of *SGSearch*, an algorithm to find *source paths* in the *Source Graph*. We then discuss a naive evaluation of the *source paths* against the *Object Graph* (O, L_O) by a mediator accessing utilities ESearch, EFetch and ELink currently supported by the NCBI.

6.1 Query Evaluation Stages

1. Validate the query against LT and LL. This corresponds to Steps 1, 2, and 3 of Sec. 5.
2. Find all *source paths* in the *Source Graph*, (S, L_S) that satisfy the query; this is Step 4. We describe an exhaustive search algorithm *SGSearch* next.
3. Eliminate meaningless *source paths* using LL. Scientists may further eliminate *source paths* that are not of interest to them, e.g., they do not use data from a specific source in the path. They may also rank the *source paths* based on domain specific criteria. Note that this step improves efficiency and usefulness but does not impact the semantics; it is not included in Sec. 5.
4. Evaluate each *source path* on the *Object Graph* (O, L_O), starting with the highest ranked *source path*. We describe a naive evaluation strategy that assumes the mediator has a *decision rule* to determine the link type of links in the *Object Graph*.
5. Return results to the user. This may include the *objects* and links of a path in the *Object Graph* or only the *target objects* reached along paths of the *Object Graph*.

6.2 SGSearch

SGSearch is an extension of a search algorithm presented in [9]. The extension is to consider labeled links in the *Source Graph*; the original algorithm only considered unlabeled links. *SGSearch* is based on a deterministic finite state automaton (DFA) that recognizes a regular expression (query Q_r). The algorithm performs an exhaustive breadth-first search of all paths that satisfy the query. *SGSearch* assumes that Q_r is semantically correct, i.e., the original query has been rewritten as described in Sec. 5 to include all needed wild card class labels, ϵ_C, and link labels, ϵ_L.

Suppose DFA is the automaton that recognizes the regular expression query Q_r. The DFA is represented by a set of transitions, where a transition is a 4-tuple t=(i,f,e,$Pred$), and where, i represents the initial state of t, f represents the final state of t, and e corresponds to the label of t. Note that $e \in C$ U L, i.e., e belongs to the set of class labels or link labels. *Pred* represents a predicate expression to be satisfied by a source that implements a class or by objects in a source. For simplicity we do not discuss *Pred* in this section. The state i (respectively f) may be a *start state* (respectively *end state*) of the DFA.

The exhaustive algorithm *SGSearch* comprises two phases: (a) *build path* and (b) *print path*. In phase *build path*, for each visited transition t^p=(i,f,e,$Pred$), *SGSearch* annotates each transaction with a set t^p.*currentImp* corresponding to the label e. If e is a class label or the wild card class label ϵ_C, t^p.*currentImp* includes *all* s_i in S such that $e \in m_S(s_i)$. If e is a link label, t^p.*currentImp* includes this label. Finally, if e is the wild card link label ϵ_L, then t^p.*currentImp* includes all the labels in L.

For each transition t^p=(i,f,e,$Pred$), if i is not a *start state* of the DFA, then *SGSearch* annotates each element n of t^p.*currentImp*, with a set n.*previousImp*; this is either s_i.*previousImp* or l.*previousImp*, depending on whether n is a source or link. If n is a link label l, *SGSearch* considers the transaction t^{p-1} previous to t^p, and creates a set l.*previousImp* that includes all sources s_j in t^{p-1}.*currentImp* that are adjacent to l in the *Source Graph*(S,L_S) and that satisfy *Pred*. These adjacent sources s_j must participate in a link, such as (s_j, l, s_i) in L_S. Note that if l does not have an adjacent source in t^{p-1}.*currentImp*, it is no longer considered as an element of t^p.*currentImp*.

If n is a source, then, *SGSearch* considers two transitions t^{p-1} and t^{p-2}, where t^{p-1} is previous to t^p and t^{p-2} is previous to t^{p-1}. It finds sources s_j in t^{p-2}.*currentImp* that satisfy *Pred* and, that are adjacent to s_i in L_S through a link label l that is included in t^{p-1}.*currentImp*. Similarly, if s_i does not have an adjacent source in t^{p-2}.*currentImp*, it is no longer considered as an element of t^p.*currentImp*.

In phase *print path*, the algorithm starts from the set t^{fin}.*currentImp* corresponding to the final transition t^{fin} whose final state is an *end state* of the DFA. For each s_i in this set, *SGSearch* uses the set s_i.*previousImp* to construct a path. The path terminates in one of the sources corresponding to the start transition t^1.currentImp; recall that the initial state of this transition is a *start state* of the DFA.

6.3 Naive Evaluation by a Mediator

We now describe a naive evaluation of the *source paths* produced by *SGSearch* against the *Object Graph.* For illustration, we consider the *NCBI Object Graph.* NCBI is the gatekeeper for NIH data sources. The Entrez utilities for search and retrieval from NCBI sources include ESearch, ELink and EFetch. Given a source and some search predicate, ESearch finds objects in the source that satisfy the predicate and EFetch retrieves those objects. Together, they act like the σ relational operator. Given an object identifier (o.UID) and a target source, ELink retrieves all links (o.UID pairs) starting from the given object o_i and reaching objects in the target source. We describe a naive evaluation strategy based on these utilities.

Repeat for each subpath (s_i, l, s_j) in a source path until the path terminates:

1. Invoke ESearch on the current source s_i with the search predicate *Pred.* Retrieve a set of object identifiers (UIDs) for some objects in O_i.

2. Invoke EFetch to obtain XML documents for each object $o_i \in O_i$. Determine those links with link label l.

3. Invoke ELink on all object links from o_i, with label l, and reaching an object o_j in source s_j. Create a set of objects O_j.

We assume that for each source link registered in L_S, the mediator has a decision rule to determine the link type of all outgoing object links from object o_i. We illustrate the decision rules using an example. Consider the portion of a UniProt entry in Fig. 8.

```
        ID    MEFV_HUMAN STANDARD; PRT; 781 AA.
        AC    O15553; Q96PN4; Q96PN5;
        DT    16-OCT-2001 (Rel. 40, Created)
        DT    16-OCT-2001 (Rel. 40, Last sequence update)
        DE    Pyrin (Marenostrin).
        ...   ...
$\rightarrow^1$  OS    Homo sapiens (Human).
$\rightarrow^1$  OX    NCBI_TaxID=9606;
        RN    [1]
        ...   ...
        CC    -!- DISEASE: DEFECTS IN MEFV ARE THE
              CAUSE OF FAMILIAL MEDITERRANEAN
        CC        FEVER (FMF) [MIM:249100]...
        ...   ...
$\rightarrow^2$  DR    MIM; 608107;
$\rightarrow^3$  DR    MIM; 249100;
        ...   ...
$\rightarrow^4$  FT    VARIANT 694 694 M -> I (in FMF).
        ...   ...
```

Figure 8: Portions of the UniProt entry O15553

The four lines marked with '$\rightarrow$' correspond to four *e-links*. We describe decision rules to classify two *e-links*. For $\rightarrow^1$, the following rule (represented by a triple) will be used to determine the link type: (./OS & ./OX, *is causal for disease*, lt_i). The first item of the triple specifies that when the two sub-elements (attributes) OS and OX occur in the UniProt entry, then this is a link of type lt_i with link label *is causal for disease.* The attribute values (OS Homo sapiens (Human) and OX NCBI_TaxID=9606) correspond to the actual object link. For $\rightarrow^4$, the following rule is used: (./FT, *genetic background*, lt_j). The attribute FT determines the rule to be of type lt_j with label *genetic background.*

7. DISCUSSION

We present the *e-link* framework of a data model and query language that allows scientists to express knowledge of links and to exploit this knowledge in answering queries. We discuss the naive evaluation of these queries by a mediator.

There are clearly many challenges that must be addressed. The first task is developing machine assisted techniques to extract semantics and to provide labels for existing links. We note that there are several ongoing efforts to enhance links [2, 3, 4, 10, 12]. This task is difficult, because a link in the *Source Graph* may often have multiple semantics. The second task is exploiting existing work in ontologies in the task of associating semantics to links. We note that in our current prototype, the semantics is limited to a simple label, whereas ontologies can support richer relationships. Finally, we have to develop robust and efficient techniques for query evaluation that scale to the large distributed *Object Graph* of the life science domain.

Acknowledgements: We thank Barbara Eckman, Stephan Heymann, Zoe Lacroix, Alex Lash, Woei-Jyh Lee and Peter Rieger for their feedback. This research was supported by NSF grants IIS0222847 and IIS0430915 and by the German Research Society (DFG grant no. NA 432).

8. REFERENCES

[1] *www.ncbi.nlm.nih.gov/entrez/query/static/elink_help.html.*

[2] *www.genome.ad.jp/dbget-bin/www_linkdb.*

[3] *www.ebi.ac.uk/integr8/.*

[4] *www.bioinf.org.uk/pdbsprotec/.*

[5] *www.geneontology.org.*

[6] *www.ebi.ac.uk/GOA/.*

[7] Barbara A. Eckman, Paul Brown, A. Kershenbaum, R. Mushlin, and S. Mitchell. The IBM DB2 systems biology graph extender research prototype. *White Paper, IBM Life Sciences*, 2005.

[8] Z. Lacroix, H. Murthy, F. Naumann, and L. Raschid. Characterizing properties of paths in biological data sources. *Proceedings of the DILS Conference and Springer-Verlag Lecture Notes in Computer Science (LNCS)*, (2994):187–202, 2004.

[9] Z. Lacroix, L. Raschid, and M.E. Vidal. Efficient techniques to explore paths in life science data sources. *Proceedings of the DILS Conference and Springer-Verlag Lecture Notes in Computer Science (LNCS)*, (2994):203–211, 2004.

[10] Alex Lash, Woie-Jyu Lee, and Louiqa Raschid. A protocol to extract and generate links capturing marker semantics from pubmed to the human genome. *Under review*, 2005.

[11] Ulf Leser. A query language for biological networks. Technical Report 187, Institut fuer Informatik der Humboldt Universitaet zu Berlin, 2005.

[12] A. Martin. PDBSprotEC: A web-accessible database linking PDB chains to EC numbers via swissprot. *Bioinformatics*, 20(6):986–988, 2004.