Pacific Symposium on Biocomputing 11:116-127(2006)

SELECTING BIOLOGICAL DATA SOURCES AND TOOLS WITH XPR, A PATH LANGUAGE FOR RDF

SARAH COHEN-BOULAKIA[†], CHRISTINE FROIDEVAUX[†] AND EMMANUEL PIETRIGA[‡]

[†]*LRI, CNRS UMR 8023, Université Paris-Sud*
91405 Orsay, CEDEX, France
{*cohen, chris*}*@lri.fr*

[‡]*INRIA Futurs, LRI*
91405 Orsay, CEDEX, France
emmanuel.pietriga@inria.fr

As the number, richness and diversity of biological sources grow, scientists are increasingly confronted with the problem of selecting appropriate sources and tools. To address this problem, we have designed BioGuide[1], a user-centric framework that helps scientists choose sources and tools according to their preferences and strategy, by specifying queries through a user-friendly visual interface. In this paper, we provide a complete RDF representation of BioGuide and introduce XPR (eXtensible Path language for RDF), an extension of FSL[2] that is expressive enough to model all BioGuide queries. BioGuide queries modeled as XPR expressions can then be saved, compared, evaluated and exchanged through the Web between users and applications.

1. Introduction

The number and size of new biological data sources together with the number of tools available for analysing this data have increased exponentially in the last few years, to a point where it is unrealistic to expect scientists to be aware of all of them. However, as these sources and tools are often complementary, focused on different objects and reflecting various experts' points of view, scientists should not limit themselves to the sources they already know well, and thus have to face the problem of selecting sources and tools when interpreting their data.

For instance, the European HKIS platform[a] offers a set of analysis scenarios where at each step users may have to ask questions necessitating the consultation of various sources. For this, we have designed the DSS algorithm[3] that builds paths allowing to navigate through data sources. DSS reflects how oncologist partners involved in the project select sources, and takes into account their preferences.

[a]www.hkis-project.com

Pacific Symposium on Biocomputing 11:116-127(2006)

Two other systems considering paths between sources and exploiting preferences have been developed in the same spirit: Biomediator[4] and Bionavigation[5]. We then wanted to investigate in a systematic way the need for various ways of querying biological sources. A thorough analysis of needs[1], by means of a questionnaire, revealed the importance of (i) expressing *transparent* queries[4] (ii) exploiting *preferences* (e.g. *reliability*) with respect to both sources and tools and (iii) following a specific *strategy*.

In response to these findings, we have designed BioGuide[1], a **user-centric framework** which helps scientists choose sources and tools according to their preferences and strategy. BioGuide has proven itself to be useful to obtain complementary data through the use of alternative ways of finding information, and to deal with divergent data by exploiting preferences. Moreover, BioGuide provides a framework which is general enough to take into account all preferences of current systems (DSS, Biomediator and Bionavigator), to simulate their behaviors by means of strategies and to specify new preferences and strategies. The BioGuide system is available for use from http://www.lri.fr/∼cohen/bioguide/bioguide.html.

BioGuide provides a simple visual interface which allows users to specify the biological entities, sources and tools they are interested in. In this paper we want to make it possible for BioGuide users to save their queries (reusability), exchange them (collaboration between experts), compare them (expressiveness) and evaluate them (efficiency). We provide a complete RDF representation of BioGuide, which is well-suited to the uniform representation of biological entities and sources structured as multi-labeled graphs. We then exploit this RDF representation of BioGuide data to model the queries expressed visually by users as queries on RDF models. For this, we introduce XPR (eXtensible Path language for RDF), an extension of FSL[2] that is expressive enough to model all BioGuide queries.

For the sake of readability, we consider in this paper a simpler version of BioGuide where strategies are not described and preferences are simplified.

2. BioGuide: selection of sources and tools

BioGuide aims at assisting users in the specification of their queries. A thorough study of how scientists consider the query process revealed that from a question expressed in natural language, they first identify the underlying biological entities and the relationships between them. For instance, in question *"On which chromosome is the BAC of my CGH array located?"*, the underlying entities are Chromosome and Bac. In BioGuide, the user

Pacific Symposium on Biocomputing 11:116-127(2006)

is supported in his task by a graphical representation of the biological domain, represented through the **entities graph** (Fig.1), in which nodes are biological entities and labeled edges are symmetric relationships. Two kinds of relationships exist: biological relationships (e.g. `causes`, `encodedBy`) and relationships achieved by tools (e.g. `similarSeq`, `mapsWith`).

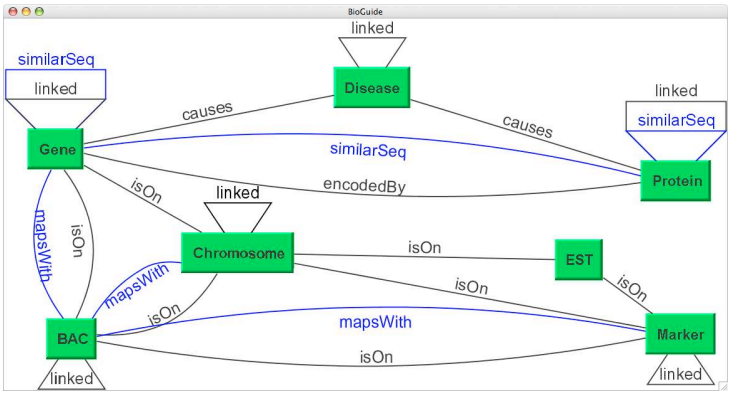

Figure 1. Entities graph fragment (BioGuide's Graphical User Interface)

Scientists can make use of this graph to build BioGuide *initial queries* by selecting entities and, possibly, relationships between these entities. It is also possible to ask the system to not only consider entities given in the query, but also to consider or avoid *additional entities* (*navigating* strategy). This can be done by explicitly referring to them or by specifying the kinds of relationships (e.g. those achieved by tools) used to reach these additional entities.

Furthermore, our study has revealed that users want to know which sources and tools can be accessed[6,7], and punctually need to cite some sources or tools. In BioGuide, they are supported in this task by a graphical representation of sources, provided by the **sources-entities graph** (Fig. 2), in which each node represents an entity in a source and arrows indicate the links between two entities (in the same source or in another). Labels on arrows specify the kind of link: cross-reference (*CrossRef*), internal link (*Internal*)– links between entities in the same source – and tools (e.g. *Blast*).

Using the sources-entities graph, scientists can thus complete their *initial query* by an *extended query* in which they (possibly) specify the sources and tools to access or to avoid. This graph is also used to visualize which

Pacific Symposium on Biocomputing 11:116-127(2006)

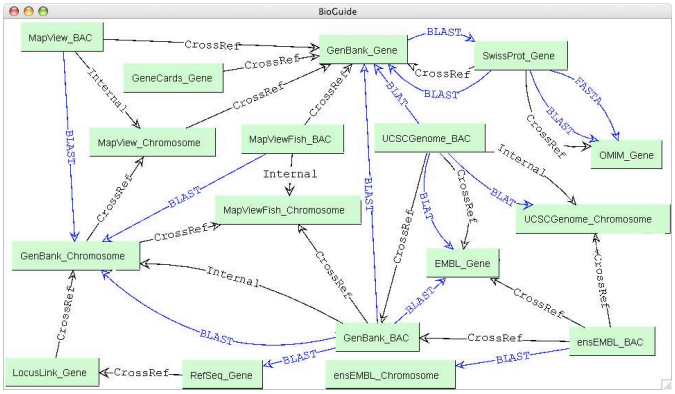

Figure 2. Sources-entities graph fragment relating BAC, CHROMOSOME and GENE

sources-entities contain a given entity and which links achieve a given relationship: the two graphs are *in correspondence* with each other. We now briefly recall BioGuide[1]'s principles and main steps (I to IV).

(I) The *initial user query* Q consists of (i) the entities and relationships underlying the user's query; (ii) the user's choice regarding the navigating strategy criterion and (iii) if necessary, the entities and kinds of relationships to avoid. (II) From Q, the *EPG* (Entity Path Generator) module yields P_e, the set of acyclic paths in the entities graph generated following the choice about the navigating strategy.

In the previous example, following the navigating strategy and avoiding the MARKER entity, the following paths are returned by *EPG*: p_1="BAC isOn CHROMO", p_2="BAC mapsWith CHROMO", p_3="BAC isOn GENE isOn CHROMO" and p_4="BAC mapsWith GENE isOn CHROMO".

In step (III) the *extended user query* Q_{se} consists of (a) P_e, the output of *EPG*, and (b) preferences (e.g. only *reliable* sources, where reliability ratings can be parameterized by each user). (IV) Using Q_{se} and the sources-entities graph, the *SEPT* (Source-Entity Path Translator) module generates the list L_{pse} of paths in the sources-entities graph which meet specified preferences and *correspond* to the paths of P_e.

SEPT first constructs the list of *entity-centered* paths, i.e., paths of cross-references or internal links between sources-entities containing some given entity. Secondly, SEPT relates paths centered on the different entities by considering the links between sources-entities which *achieve* the relationships specified in the paths of entities. All possible paths of sources-entities are generated as alternative ways of finding information. In our

Pacific Symposium on Biocomputing 11:116-127(2006)

previous example, the following two paths (among others) are returned by *SEPT*: (MapView,BAC) $\stackrel{Internal}{\rightarrow}$(MapView,CHROMO) (corresponding to p1) and (GenBank,BAC) $\stackrel{Blast}{\rightarrow}$ (RefSeq,GENE) $\stackrel{CrossRef}{\rightarrow}$ (LocusLink, GENE) $\stackrel{CrossRef}{\rightarrow}$ (GenBank, CHROMOSOME) (corresponding to p_4).

BioGuide thus provides users with alternative *ways of finding data* based on an internal data model that allows them to specify powerful graph queries involving entities, sources and tools through a simple visual interface. More information about the architecture and data model is available[1].

Our goal is now to make it possible for BioGuide users to save their queries, exchange them with other users, compare them and evaluate them on graphs containing different components or based on different settings. BioGuide data structures thus need to be represented within a uniform, open and extensible format and queries expressed within a formal language.

3. A framework for exchanging BioGuide data

Most formats for data interchange between users over the Web and between heterogeneous applications are now based on XML. But as Semantic Web technologies mature, languages such as RDF become more attractive, further increasing interoperability and knowledge exchange capabilities, as well as allowing autonomous software agents to exploit and reason on data that is marked up semantically with RDFS/OWL ontology-based vocabularies.

3.1. *Modeling BioGuide in RDF*

RDF and its companion languages[b] offer a general-purpose framework for representing information about Web resources with domain-specific vocabularies in a minimally constraining yet uniform way. They thus represent a good candidate solution for promoting data integration from Web biological sources. Furthermore, RDF's data model[8] fits naturally with BioGuide's as both are based on directed labeled graphs (as opposed to XML's tree-based data model), making the mapping between BioGuide's data structures and RDF straightforward.

RDF describes Web *resources* identified by their URI (*Uniform Resource Identifier*) in terms of property-value pairs representing relationships between resources and characteristics of these resources. An RDF model is a collection of statements taking the form of (*subject*, *predicate*, *object*) triples. This set of statements can be represented as a directed labeled graph[8]. In

[b]http://www.w3.org/RDF/

Pacific Symposium on Biocomputing 11:116-127(2006)

the remainder of this paper we take a graph-centric view of RDF models in which *nodes* are resources (depicted as ovals) or literal values such as strings or integers (depicted as rectangles), and *arcs* are properties. Arcs are labeled by a URI identifying the property's type. Property types and resource classes (resources can be stated to be instances of classes with property `rdf:type`) are defined in RDF schemas using RDFS, RDF's vocabulary description language.

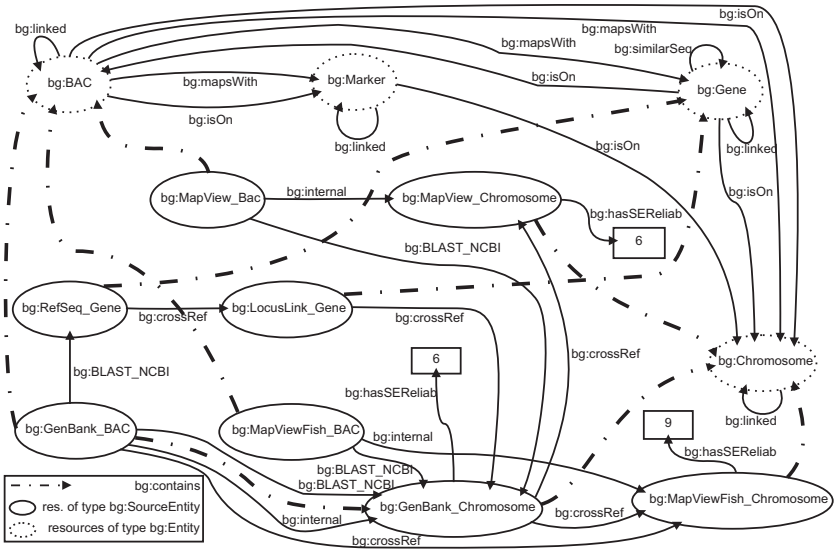

Figure 3. Fragment of BioGuide's RDF model relating various entities (schematic repr.)

Data from both graphs (Fig. 1 and 2) is modeled in a single RDF graph (Fig. 3), as elements of the two graphs are put in correspondence using property `bg:contains` (see e.g. `bg:LocusLink_Gene` and `bg:Gene`). Preferences associated with sources-entities are modeled with properties such as `bg:hasSEReliab` pointing to literal values (e.g. giving access to the level of Reliability of a given Source-Entity). There is an inverse property for each `bg:mapsWith` and `bg:isOn` property; these have been removed from the figure for legibility purposes.

Each RDF vocabulary being identified by a namespace, we introduce the BioGuide namespace whose URI is bound to prefix `bg` in this paper. Usual prefixes are bound to common namespaces: `rdf` and `rdfs` for the RDF and RDFS vocabularies. Figure 4 contains a subpart of the RDF

Pacific Symposium on Biocomputing 11:116-127(2006)

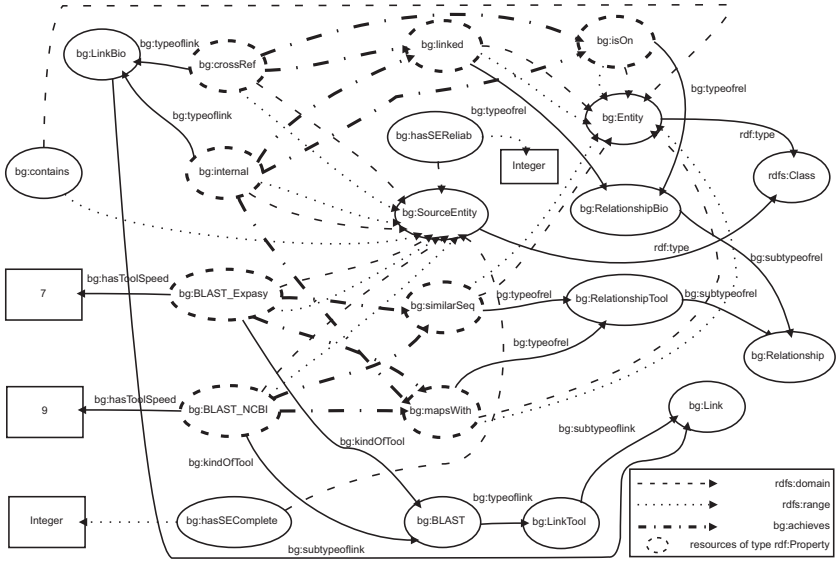

Figure 4. Fragment of BioGuide's RDF schema (schematic representation)

schema describing BioGuide's vocabulary. The schema defines all elements of the BioGuide namespace: properties (e.g `bg:contains`, `bg:internal`) and classes of resources (e.g. `bg:Entity`, `bg:SourceEntity`) that are then used in BioGuide's RDF model (Fig. 3). All properties are defined as resources in the schema graph and each have a *domain* and *range*, indicating which class of resources and typed literal values they can connect. The schema also contains BioGuide-specific, non-RDFS meta-properties, which characterize BioGuide properties in a model-wide manner: `bg:typeOfLink` and `bg:typeOfRel` classify types of links and relationships; `bg:achieves` specifies which relationship(s) a given link achieves; other meta-properties such as `bg:hasToolReliab` are used to indicate preferences values.

3.2. *Exploiting BioGuide's RDF representation*

Our goal is to exploit the RDF representation of BioGuide data to model the queries – expressed visually by users through BioGuide's user interface[1] – as queries on RDF models. BioGuide queries being essentially traversal paths in graphs, we are interested in a language for modeling paths in RDF graphs. Such a language should make it possible to express filtering conditions on nodes and arcs, and provide means of exploiting the meta-

Pacific Symposium on Biocomputing 11:116-127(2006)

information contained in schemas associated with instance data (subclass relationships, characterization of the kinds of links, tool preferences, etc.).

Most of the existing RDF query languages (e.g. RQL[9], SPARQL[10]) offer essential features such as schema/ontology awareness and limited inference capabilities, but are not well-suited to modeling traversal paths in RDF graphs. ρ-Queries[11] consider pairs of RDF resources, identify complex relationships between them, and return those relationships as property sequences. Although such sequences can be seen as paths, these are query results, i.e., instances of paths in the graph; ρ-Queries themselves cannot model queries as traversal paths.

Several proposals for RDF path languages have been made, but almost none of them is fully and formally specified: the proposals are often drafts giving short descriptions of the language constructs and a few example queries. One exception is FSL[2], a fully specified language for modeling traversal paths in RDF graphs, designed to address the specific requirements of a selector language for Fresnel[12] (an RDF vocabulary for modeling RDF data presentation knowledge). Trying to avoid reinventing the wheel, FSL is inspired by XPath[13] (W3C's language for expressing traversal paths in XML trees), reusing many of its concepts and syntactic constructs while adapting them to RDF's graph-based data model. In FSL, RDF models are considered as directed labeled graphs according to RDF Concepts and Abstract Syntax[8]. FSL is therefore fully independent from any serialization of RDF and meets many of our requirements. However, it is designed for use in the context of Fresnel and tries to be as simple as possible (its goal is not to be a full so-called RDFPath language). It is therefore not expressive enough to model all BioGuide queries. In the next section we introduce a proposal for an extension of FSL called XPR (eXtensible Path language for RDF) that addresses this problem.

4. XPR

4.1. *Extending FSL*

An FSL expression represents a path from a node or arc to another node or arc, passing by an arbitrary number of other nodes and arcs. As in XPath, steps are separated by the slash symbol (e.g. `nodeStepA/arcStepB/nodeStepC`). Each step on the path, called a *location step*, follows the XPath location step syntax: `AxisSpecifier::Test[Predicates]` where `AxisSpecifier` is an optional axis declaration specifying the traversal direction in the directed graph, `Test` is

Pacific Symposium on Biocomputing 11:116-127(2006)

a type test taking the form of a URI reference represented as an XML qualified name (QName), or a `*` when the type is left unconstrained, `Predicates` is an optional list of further filtering conditions on the nodes or arcs to be matched by this step. The type test constrains property arcs to be labeled with the URI represented by the QName, and resource nodes to be instances of the class identified by this QName (found in the schema). In other words, type tests specify constraints on the types of properties and classes of resources to be traversed and selected by paths. Constraints on the URI of resources can be expressed as predicates associated with node location steps using function call `uri(.)`. As in XPath, a dot represents the node or arc considered by the current location step. The following example models paths starting at the resource identified by `bg:BAC` (with no constraint on its type), going through arcs labeled by `bg:mapsWith` and ending at any resource of type `bg:Entity` (e.g. `bg:Gene` and `bg:Marker` in Fig. 3).

$$*[uri(.)=\text{"}bg{:}BAC\text{"}]/bg{:}mapsWith/bg{:}Entity$$

More information about FSL, including its grammar, data model and semantics, as well as examples, can be found on the FSL Web page[2].

A limitation of the FSL language is that it does not provide an equivalent of XPath's `descendant` axis (often abbreviated `step1//step2`) that specifies an unconstrained number of elements between `step1` and `step2` (possibly equal to zero). Such a construct adds significant complexity to a path language for RDF and its implementations, mainly because of the possibly cyclic nature of RDF graphs. Considered too costly by the designers of FSL with respect to its added value in the context of Fresnel, the construct is however useful from a more general RDF perspective and is mandatory to express BioGuide queries. XPR extends FSL and allows the use of the `//` notation to specify an unconstrained number of arc and node steps between two explicit location steps[c]. As shown in the expression below, it is possible to express constraints on the nodes and arcs traversed between those two explicit steps: predicates before the semi-colon are evaluated against nodes, predicates after the semi-colon against arcs. Concrete examples of use of this extension are given in section 4.2.

$$step1//_{[nodePredicateExpr;arcPredicateExpr]}step2$$

The second extension to FSL made by XPR is a new function named `rp` (for reify property). Given a property arc, `rp` returns the resource typed

[c]As BioGuide queries implicitly state that paths cannot traverse the same arc twice, the closure mechanism defined by XPR to handle cycles in RDF graphs is not detailed here.

Pacific Symposium on Biocomputing 11:116-127(2006)

as `rdf:Property` that represents this property in the corresponding RDF schema. For instance, the following **arc** location step:

$$*[rp(.)/bg{:}achieves/bg{:}isOn]$$

selects all property arcs which are stated to achieve `bg:isOn` in the RDF schema. Applied to the graph in Figure 3, it would select all `bg:crossRef` and `bg:internal` arcs since in the RDF schema of Figure 4 only those two properties are stated to achieve `bg:isOn`.

XPR has been defined as a general purpose extension to FSL which does not make any assumption about the first location step's nature. In the context of BioGuide, we only consider XPR path expressions starting with a node location step, and thus remove any ambiguity about the nature of the first location step. The result of evaluating an XPR expression on a graph is the set of all node/arc sequences in this graph that are instances of the path described by the XPR expression.

4.2. *BioGuide queries as XPR expressions*

The following example illustrates how BioGuide queries can be easily expressed as XPR path expressions, and how such expressions evolve through the various steps of the BioGuide query process. We first specify a query introduced in section 2: *"On which* Chromosomes *are* Bacs *located?"*, with the additional constraint that intermediate entities cannot be Marker, but without restriction on the kind of relationships to avoid.

$$*[uri(.) = "bg{:}BAC"]//_{[uri(.)!="bg:Marker";*]}*[uri(.) = "bg{:}Chromosome"]$$

From this query, the EPG module generates instantiated paths in the entities graph, such as `bg:BAC/bg:isOn/bg:Chromosome` (Fig. 3). Each of these result paths is then rewritten in terms of query paths involving the *corresponding* sources-entities with additional constraints based on user preferences about the sources and tools to be selected.

As an example, the following expression specifies that source *GenBank* should not be considered for entity Bac, and that only *reliable* sources (e.g. reliability higher than 7) should be considered for entity Chromosome.

$$*[bg{:}contains/*[uri(.) = "bg{:}BAC"] \text{ and } uri(.) != "bg{:}GenBank\_BAC"] \quad (1)$$

$$//[*[bg{:}contains/*[uri(.) = "bg{:}BAC"]] \text{ and } uri(.) != "bg{:}GenBank\_BAC"; \quad (2)$$
$$*[uri(.) = "bg{:}crossRef" \text{ or } uri(.) = "bg{:}internal"]]$$

$$*[arc(.) \text{ and } rp(.)/bg{:}achieves/*[uri(.) = "bg{:}isOn"]] \quad (3)$$

$$//[*[bg{:}contains/*[uri(.) = "bg{:}Chromosome" \text{ and } bg{:}hasReliab/text() > 7]]; \quad (4)$$
$$*[uri(.) = "bg{:}crossRef" \text{ or } uri(.) = "bg{:}internal"]]$$

$$*[node(.) \text{ and } bg{:}contains/*[uri(.) = "bg{:}Chromosome" \text{ and } bg{:}hasReliab/text() > 7]] \quad (5)$$

Pacific Symposium on Biocomputing 11:116-127(2006)

This single XPR expression has been split for better legibility: lines (1), (3) and (5) are location steps and double lines (2) and (4) are `//` subpaths with conditions on nodes and arcs respectively on the first and second line. Lines (1) and (5) as well as `//` subpaths (2)(4) model (possibly empty) *entity-centered* paths, i.e., paths of sources-entities containing a given entity (first BAC, then CHROMOSOME), going through cross-references or internal links, and meeting reliability preferences. Function `rp` is used (3) to relate *entity-centered* paths by selecting only links which achieve relationship `bg:isOn`.

Lastly, the SEPT module generates paths between sources-entities as alternative ways of finding data, such as (Fig. 3):

```
bg:MapviewFish_BAC/bg:internal/MapviewFish_Chromosome
```

User queries, intermediate results and final output of BioGuide are thus expressed in a single unified modeling framework based on XPR.

5. Conclusion

We have defined a complete RDF representation of BioGuide data and introduced XPR, an RDF path language extending FSL[2]. As a general-purpose path language for RDF, we expect XPR to be of interest in many application domains[12,14,15,16]. In the context of BioGuide, XPR makes it possible to express queries involving biological entities, source-entities, kinds of links and preferences in a uniform way, and to simulate the *navigating* strategy. The language is thus used to model the queries expressed visually by scientists through the system's graphical interface (users will never have to deal directly with XPR path expressions), allowing them to save, evaluate, exchange and possibly publish on the Web BioGuide queries and query results.

We are currently studying how BioGuide queries can be compared by building XPR patterns, and how to obtain statistics about them (e.g. find out the most frequently asked source). We are also developing a module to enable the use of BioGuide on top of the well-known SRS[17] platform in order to automatically retrieve instances corresponding to elements of the paths generated by BioGuide.

References

1. Cohen-Boulakia, S., Davidson, D., Froidevaux, C.: A User-centric Framework for Accessing Biological Sources and Tools. *To appear in Proc. DILS, Data Integration for the Life Sciences*, Springer-Verlag, Lecture Notes in Computer Science (LNCS) series. (2005).

2. Pietriga E.: Fresnel Selector Language for RDF (FSL), (2005) *http://www.w3.org/2005/04/fresnel-info/fsl/*
3. Cohen-Boulakia, S., Lair, S., Stransky, N., Graziani, S., Radvanyi, F., Barillot, E., Froidevaux, C.: Selecting biomedical data sources according to user preferences, *Bioinformatics, Proc. ISMB/ECCB04*, **20**, i86-i93 (2004).
4. Shaker, R., Mork, P., Brockenbrough J.S., Donelson L., Tarczy-Hornoch P.: The BioMediator System as a Tool for Integrating Biologic Databases on the Web. *Proc. VLDB Workshop on Information Integration on the Web* (2004).
5. Lacroix, Z., Raschid, L., Vidal, M.: Efficient Techniques to Explore and Rank Paths in Life Science Data Sources, *Proc. Data Integration in the Life Sciences*, 187-202 (2004).
6. Buneman, P., Khanna, S., Tan, W.: Why and Where: A Characterization of Data Provenance, *Proc. Int. Conf. on Database Theory (ICDT)*, 316-330 (2001).
7. Zhao, J., Wroe, C., Goble, C., Stevens, R., Quan, D. and Greenwood, M.: Using Semantic Web Technologies for Representing e-Science Provenance *Proc Semantic Web Conference (ISWC)*, 92-106 (2004).
8. Klyne G., Carroll J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, (2004) *http://www.w3.org/TR/rdf-concepts/*
9. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M: RQL: A Declarative Query Language for RDF. *Proc. World Wide Web conference (WWW)* (2002).
10. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF, W3C Working Draft, (2005) *http://www.w3.org/TR/rdf-sparql-query/*
11. Anyanwu, K., Seth A.: ρ-Queries: Enabling Querying for Semantic Associations on the Semantic Web. *Proc. World Wide Web Conf. (WWW)* (2003).
12. Bizer C., Lee R., Pietriga E.: Fresnel - A Browser-Independent Presentation Vocabulary for RDF, *Proceedings of the Second International Workshop on Interaction Design and the Semantic Web* (2005)
13. Clark J., DeRose S.: XML Path Language (XPath) Version 1.0, (1999) *http://www.w3.org/TR/xpath*
14. Angeles, R., Gutierrez, C.: Querying RDF Data from a Graph Database Perspective, *To appear in Proc. Europ. Semantic Web Conf. (ESWC)* (2005).
15. Pietriga E.: Styling RDF Graphs with GSS, XML.com, (2003) *http://www.xml.com/pub/a/2003/12/03/gss.html*
16. Haase, P., Broekstra, J., Eberhart, A. and Volz, A.: A comparison of RDF query languages. *Proc. Int. Semantic Web Conference (ISWC)* (2004).
17. Etzold, T., Ulyanov, A. and Argos, P.: SRS: information retrieval system for molecular biology data banks. *Methods Enzymol*, **266**, 114-128 (1996).