SMART: A Web-Based, Ontology-Driven, Semantic Web Query Answering Application

Alexander De Leon Battista[1], Natalia Villanueva-Rosales[1], Myroslav Palenychka[1], Michel Dumontier[1,2]

[1] School of Computer Science, [2]Department of Biology,
Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, K1S5B6 Canada
{adlbatti,nvillanu,mpalenyc}@scs.carleton,ca, michel_dumontier@carleton.ca

Abstract. SMART (Semantic web information Management with automated Reasoning Tool) is an open-source project, which aims to provide intuitive tools for life scientists for represent, integrate, manage and query heterogeneous and distributed biological knowledge. SMART was designed with interoperability and extensibility in mind and uses AJAX, SVG and JSF technologies, RDF, OWL, SPARQL semantic web languages, triple stores (i.e. Jena) and DL reasoners (i.e. Pellet) for the automated reasoning. Features include semantic query composition and validation using DL reasoners, a graphical representation of the query, a mapping of DL queries to SPARQL, and the retrieval of pre-computed inferences from an RDF triple store. With a use case scenario, we illustrate how a biological scientist can intuitively query the yeast knowledge base and navigate the results. Continued development of this web-based resource for the biological semantic web will enable new information retrieval opportunities for the life sciences.

Keywords: Semantic Web, Query Answering, Graphical User Interface.

1 Introduction

The primary goal of the Semantic Web is to add semantics to the current Web in such a way that machines can reason about the content [1]. Research activities in the Dumontier Lab are focused on using semantic web technologies to enable biochemical knowledge discovery [2, 3]. In combination with complimentary efforts, we expect that more powerful knowledge discovery tools will facilitate data integration in bioinformatics and create powerful new data mining opportunities over heterogeneous biochemical knowledge. However, friendly interfaces to query over the semantic web remain a critical limitation in their use and adoption.

The open-source Semantic web information Management with automated Reasoning Tool (SMART) project aims to provide intuitive interfaces for the *representation*, *integration*, *management* and *querying* of knowledge using the semantic web. In this paper, we describe an initial web-based application that adopts an ontology centric model to perform semantic query answering over heterogeneous yeast biological knowledge. Ontologies in this application are described using OWL DL, a sublanguage of the Web Ontology Language (OWL) [4]. Features of the application include semantic query composition and validation using DL reasoners, a graphical representation of the query, a mapping of DL queries to SPARQL[1] and the retrieval of pre-computed inferences from an RDF triple store.

2 System Architecture

The system architecture shown in Fig. 1 is composed of several components including an application mediator, a web-based user interface, an ontology repository, an ontology index, a hybrid reasoner which optimizes the reasoning process by converting a DL expression into SPARQL, and a SPARQL query engine for querying a database/triple store. SMART was implemented using the Java programming language and interoperates with other open source and Semantic Web technologies, including the WonderWeb OWL API[2], Pellet[3], Jena[4], and Protégé[5].

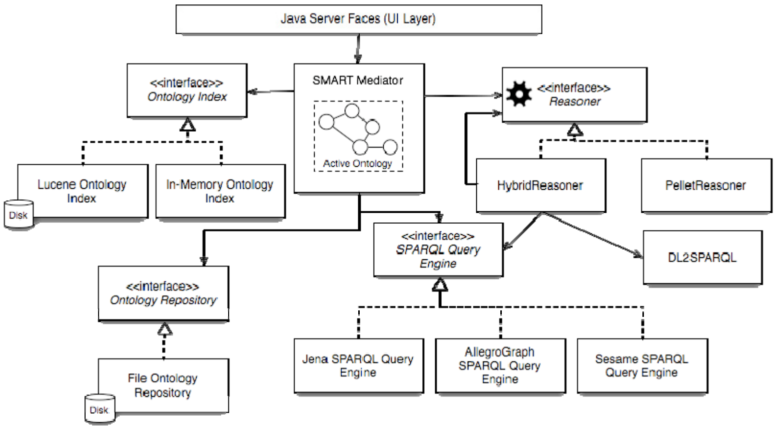


Fig. 1. SMART Architecture

SMART Mediator: The SMART Mediator is the core component of the system which enables decoupled interactions among components, controls the behavior of the system, and manages ontologies used by the application. The mediator loads URI-identified ontologies via the ontology repository. The retrieved ontology becomes the active ontology upon which reasoning and query answering may occur. The mediator will also resolve and load ontologies imported by the active ontology. The mediator enables communication between components and this supports sophisticated multi-component behavior such as the use of a reasoner or ontology index by the user

[1] http://www.w3.org/TR/rdf-sparql-query/
[2] http://wonderweb.man.ac.uk/owl/
[3] http://pellet.owldl.com/
[4] http://jena.sourceforge.net/
[5] http://protege.stanford.edu/

interface. The manipulation of OWL documents and data structures by multiple SMART components is facilitated by the OWL API.

Ontology Repository: The ontology repository supports storage and retrieval of URI-identified ontologies. The current implementation will store new or modified ontologies into the repository. Ontologies can be stored and retrieved in/from the repository by using its URI or the combination of the URI and a version number, thus it provides a simple mechanism for versioning modified ontologies. This implementation uses a file based system to store ontologies, but other implementations could use more sophisticated storage solutions such as relational databases or version control systems.

Ontology Indexer: The ontology indexer enables fast and easy access to entities defined in the active ontology. Indexed entities include class, property, individual names and their types. Thus, this component supports partial entity name matches, and retrieval of specific types of entities. The current implementation leverages the speed and power of the Lucene[6] Text Indexer, therefore, it eliminates the need for in-memory data structures for caching entity metadata.

Hybrid Reasoner: Reasoning about large ontologies can be expensive both in terms of time and space. For this reason, we have implemented a "cached-reasoning" service that retrieves pre-computed inferences about an OWL-DL ontology. Ontologies loaded into the system are sent to a DL reasoner offline and the inferences retrieved are explicitly stored in the triple store. The Hybrid Reasoner component uses the DL2SPARQL component to convert an OWL Class expression to a SPARQL query which is sent to a SPARQL query engine via the SMART mediator. The hybrid reasoner currently answers queries that involve subsumption, realization and role restrictions. Queries composed of negation or cardinality restrictions currently require a DL reasoner, or a knowledge base that integrates a DL reasoner. Nevertheless, this component optimizes the querying of large OWL ontologies that could take a long time to reason about or would require more memory than is currently available.

SPARQL Query Engine: The SPARQL Query Engine provides the ability to query a triple store repository using SPARQL. The interface provides an API for components that require SPARQL query services, such as sending a SPARQL query and obtaing a result set. Implementations for Jena, AllegroGraph and Sesame serve as wrappers for the vendor-specific APIs, hiding them from the components that use the services.

Graphical User Interface: SMART's web-based, graphical user interface supports semantic query construction and validation, result retrieval and presentation. The graphical interface is built primarily on the combination of two important technologies: Java Server Faces (JSF)[7] and Asynchronous Javascript and XML (AJAX)[8]. This combination supports the development of reusable event-driven UI

[6] http://lucene.apache.org/

[7] http://java.sun.com/javaee/javaserverfaces/

[8] http://ajax.asp.net/

components connected to a server application that eliminates the need to reload pages. Such behavior was desirable as certain components were expected to be re-used by different parts of the user interface, and in which the ability to react to user content in real time was also deemed essential. The user interface has been tested and operates optimally with Mozilla FireFox 1.5.

3 Semantic Query Answering

We believe that semantic query answering occurs when ontologically-defined entities and roles are used to answer conjunctive queries. In this way, ontologies define the lexical granularity from which queries may be formulated and offer the necessary semantics to evaluate whether an answer to a query may possibly exist.

A query language will determine the possible questions that a user can ask to the system. For example, in the relational database world, SQL is the standard language used for creating queries. Several rich query languages exist for querying RDF/OWL data. These include RQL [5, 6], nRQL [7]. While powerful queries may be formulated using these languages, they require training and thus pose a challenge for non-experts users. To facilitate semantic query formulation, we utilize the Manchester OWL Syntax [8] to define *DL queries* (i.e. queries defined by a class description). An important feature of this language, which makes it more readable and easier to understand than traditional descriptive logics syntax, is that it uses an infix notation rather than a prefix notation for keywords. E.g. the traditional DL expression: Father $\cap$ $\exists$ isMarriedTo.Architect would be written as “Father that isMarriedTo some Architect” in this syntax. While room for improvement still exists, constructing class expressions using this syntax is substantially easier to formulate by non-DL experts, and reduces the technical barrier of learning another query language.

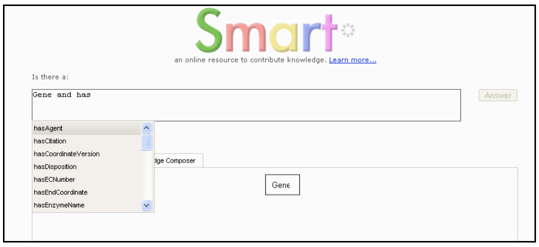
In SMART, DL queries are created in a text box using the web-based user interface. Real-time feedback allows users to construct syntactic, semantic and logical valid queries. This feedback is given as follows: First, a list of entities contained in the knowledge base (class, property, and individual names) are suggested based on a partial word matches as they are entered by the user. The user may continue typing the word or select one of the suggested words. However, if the user types a word that is not found in the system, an error appear as the query is not valid. Moreover, the correct type of entity is suggested for the phrase under construction thereby encouraging users to build grammatically valid queries. Second, a valid DL query is verified by a DL reasoner on the server side, and a message appears below the query expression which notifies, as the user enters the query, if his/her class expression is valid or not. This allows early recognition of semantic errors, and enforces logically valid expressions. Finally, a graphical representation of the query is displayed to aid in the interpretation of complex or nested expressions. These dynamic features were made possible by leveraging the Manchester OWL Syntax parser from Protégé 4 (*alpha*) that convert the DL query into an OWL API object that can be exchanged with a DL reasoner.

Use Case Scenario: Querying Yeast Biological Knowledge

This section aims to illustrate how SMART may be used for semantic query answering over yeast biological knowledge. We loaded the yOWL ontology and 3,423 instances populated from the Saccharomyces Genome Database (SGD). For more information about this ontology, refer to [2]. The yOWL ontology currently contains 244 classes, from which 38 are defined, 57 object properties and 22 annotation properties[9].

In this scenario, a scientist wants to know which yeast genes/proteins that exhibit transferase activity (aka transferases) are part of a molecular complex and have been associated with a nonviable (or fatal) outcome when removed.

Our scientist will first access SMART online at http://smart.dumontierlab.com and begin composing her query in the query text box. She first tries “transferase”, but SMART offers no matching terms in the suggestion box. She decides to make her query more explicit by defining exactly what she means. She starts by typing “Gene”, SMART suggests this word among others based on partial word matching to knowledge base entities. She selects “Gene” by hitting the tab button, presses the space bar, and begins typing a character. A message indicates that the expression is not valid and when she presses the “why?” link, a window appears to explain that SMART expects a boolean class constructor (and, or, not, that) to formulate a conjunctive query. She quickly types the “and” operator.



an online resource to contribute knowledge. Learn more...

Is there a:

Gene and has

Fig. 2. SMART web-based interface for query composition with suggestions.

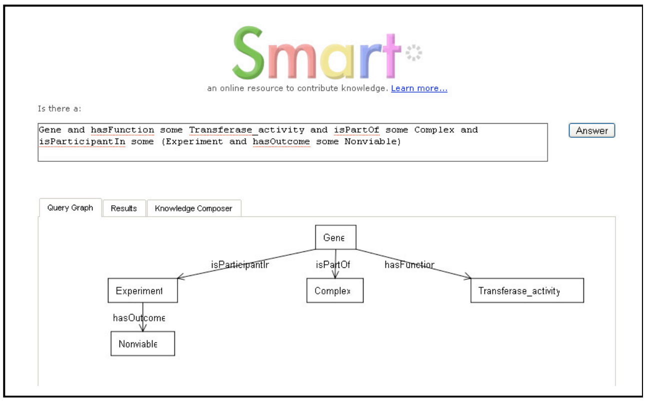
Next, she would like to look for genes that have the transferase function as defined by the Gene Ontology (GO)[10]. Fig. 2 illustrates the property suggestions starting with the letters “has”. She selects “hasFunction” and is now prompted to select existential/universal or cardinality restrictions (some, only, value, min, max, exactly) to modify the range of the predicate. Knowing that genes/proteins often have multiple functions, she wants to find “some” that have at least this function and defines the property range as the GO term *Transferase_activity*. She formulates the following syntactically, logically and semantically correct DL query to her question:

Gene and hasFunction some Transferase_activity and isPartOf some Complex and isParticipantIn some (Experiment and hasOutcome some Nonviable)

[9] http://ontology.dumontierlab.com/yOWL-1.0

[10] http://www.geneontology.org/

A corresponding graphical representation of the text query can simultaneously be seen in the *Query Graph* tab (Fig. 3). This graph query will allow our scientist to visualize the query in a more intuitive way. Once the user is satisfied with the query, she can then click the *Answer* button to obtain the results.



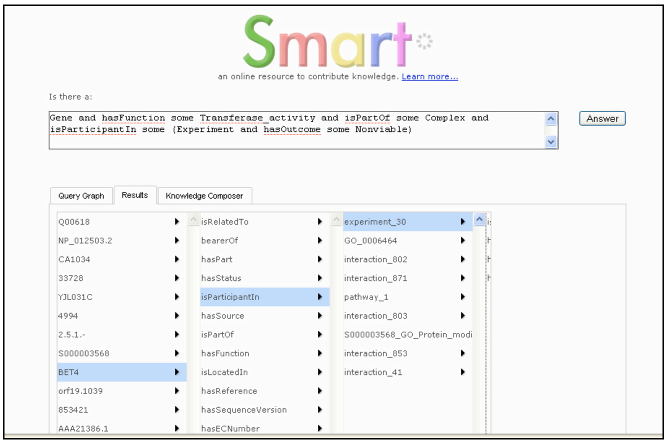
```
Gene and hasFunction some Transferase_activity and isPartOf some Complex and
isParticipantIn some (Experiment and hasOutcome some Nonviable)
```

Fig. 3. SMART web-based interface screenshot showing the synchronization between the text query and the query graph generated.

This query will now be answered by the hybrid reasoner which maps the DL query to the following SPARQL query to be answered, in this case, by Jena:

```
SELECT DISTINCT ?i
WHERE {{{
{?i <http://ontology.dumontierlab.com/bro-.0#isParticipantIn> _:var2.
 _:var2 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://ontology.dumontierlab.com/yowl-1.0#Experiment>.
_:var2  <http://ontology.dumontierlab.com/yowl-.0#hasOutcome> _:var3.
_:var3  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://ontology.dumontierlab.com/yowl-1.0#Nonviable>}}
{?i  <http://ontology.dumontierlab.com/bro-.0#isPartOf> _:var0.
_:var0  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://ontology.dumontierlab.com/yowl-1.0#Complex>.
?i  <http://ontology.dumontierlab.com/bro-1.0#hasFunction> _:var1.
_:var1  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://ontology.dumontierlab.com/goslim-1.0#Transferase_activity>.
?i  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://ontology.dumontierlab.com/yowl-1.0#Gene>}}}
```

The result is then displayed in the Results tab (Fig. 4), which supports HTML navigation of RDF/OWL graphs by dynamically retrieving individuals (nodes) their associated properties (edges) from the knowledge base. For instance, on selecting the “BET4” protein, a new panel to the right of the one selected lists properties associated to “BET4”. On selecting the “isParticipantIn” property, a new panel opens to the right of the property panel. By selecting one of the results such as “experiment_30”, she may further explore what is know about “experiment_30” or backtrack to explore other individuals. Thus, the path across multiple related individuals can be kept while providing the flexibility to investigate other individuals and their relations.



Is there a:

```
Gene and hasFunction some Transferase_activity and isPartOf some Complex and
isParticipantIn some (Experiment and hasOutcome some Nonviable)
```

Q00618
NP_012503.2
CA1034
33728
YJL031C
4994
2.5.1.-
S000003568
BET4
orf19.1039
853421
AAA21386.1

isRelatedTo
bearerOf
hasPart
hasStatus
isParticipantIn
hasSource
isPartOf
hasFunction
isLocatedIn
hasReference
hasSequenceVersion
hasECNumber

experiment_30
GO_0006464
interaction_802
interaction_871
pathway_1
interaction_803
S000003568_GO_Protein_modi
interaction_853
interaction_41

Fig. 4. SMART web-based interface screenshot showing the results navigation.

4 Discussion

SMART empowers users to create simple expressions that allow them to pose sophisticated conjunctive queries over a knowledge base. Query answering operates under the open world assumption and the lack of unique name assumption according to the Semantic Web semantics. The results obtained from these queries may include class membership inferences (using necessary and sufficient conditions to define classes) that are not available from browsing the original SGD database. Moreover, capturing user-generated definitions (i.e. transferase as defined in the use case) will create opportunities for capturing ontological concepts with more powerful semantics than with Web 2.0 tagging. SMART can handle any ontology in the OWL language for semantic query answering. To overcome problems with on the fly reasoning about a large knowledge base, we used pre-computed inferences using a DL reasoner. This substantially speeds up query answering on typical queries involving class subsumption and class membership over a large amount of distributed and heterogeneous biological data. However, substantial challenges remain for truth maintenance upon the addition of new knowledge.

A general design principle of the user interface, based on results obtained from a recent survey carried out in our lab for Canadian scientists, was to investigate the use of English-like phrases for query composition. Surveys on the ease of use and functional aspects by biology and computer science students in the lab have yielded positive feedback, although a more rigorous usability study has yet to be performed. While the user-tested Manchester OWL syntax and the Protégé expression parser lend a first step towards this directly, it is anticipated that other, more sophisticated controlled natural languages such as Sydney OWL syntax [9] will play a major role as an enabling technology. In this work, we also explored the use of a graphical language (similar to GLOO [10]) for simplifying the interpretation of complex (and/or) nested expressions. We expect that the addition of interactive elements to our graphical graph query interface will enable an interesting and alternative mechanism for query formulation that might appeal to a large segment of the public. Since a user never has to read/write/understand XML/RDF/OWL, SPARQL, SQL or any other query language created for machine consumption, we consider that further developments in this area is essential for non-experts to tap into the potential of the semantic web.

5 Conclusions and Future Work

In this paper, we described the modular architecture and basic functionality of SMART, an easy to use web-based application for the semantic web. Initial functionality includes semantic query answering and browsing the results over an OWL knowledge base using (when possible) pre-computed inferences. Ongoing work includes exploratory content browsing, interactive graphical query formulation, knowledge composition, and truth maintenance. Efforts are also being directed to support user-specified ontology imports, queued reasoning, and implementation of DL extensions for SPARQL.

Acknowledgments. We would like to thank Kirill Levinsky, Faris Matari and Anne Taylor for technical contributions and useful discussions. This work was made possible by Luc Lalande and funding from Foundry Program for AL and Talent First for MP. Funding for NVR and MD was with CONACYT scholarship #150581 and NSERC, respectively.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. (2001)
2. Villanueva-Rosales, N., Osbahr, K., Dumontier, M.: Towards a Semantic Knowledge Base for Yeast Biologists. HCLS-DI (2007)
3. Villanueva-Rosales, N., Dumontier, M.: Describing chemical functional groups in OWL-DL for the classification of chemical compounds. OWLED (2007)
4. W3C: OWL Web Ontology Language Guide. In: Smith, M.K., Welty, C., McGuinness, D.L. (eds.) (2004)
5. Haarslev, V., Möller, R., Wessel, M.: Querying the Semantic Web with Racer + nRQL. KI-04 Workshop on Applications on Description Logics, Ulm (2004)
6. Gregory, K., Sofia, A., Vassilis, C., Dimitris, P., Michel, S.: RQL: a declarative query language for RDF. Proceedings of the 11th international conference on World Wide Web. ACM Press, Honolulu, Hawaii, USA (2002)
7. Fikes, R., Hayes, P., Horrocks, I.: OWL-QL--a language for deductive query answering on the Semantic Web. Web Semant. Vol. 2 (2004)
8. Horridge, M., Drummond, N., Goodwin, J., Rector, A., Stevens, R., Wang, H.H.: The Manchester OWL Syntax. OWLED 2006, Athens, Georgia (2006)
9. Fuchs, N.E., Schwitter, R.: Web-Annotations for Humans and Machines. ESWC (2007)
10. Fadhil, A., Haarslev, V.: GLOO: A Graphical Query Language for OWL Ontologies. OWLED 2006 (2006)