Developing a Web-Based Application using OWL and SWRL

Martin J. O'Connor, Ravi Shankar, Csongor Nyulas, Samson Tu, Amar Das

Stanford Medical Informatics,
Stanford University, Stanford, CA 94305-5479
{martin.oconnor, ravi.shankar, csongor.nyulas, swt, das}@stanford.edu

Abstract

Data integration is central in Web application development because these applications typically deal with a variety of information formats. Ontology-driven applications face the additional challenge of integrating these multiple formats with the information stored in ontologies. A number of mappings are required to reconcile the variety of formats to produce a coherent overall system. To address these mappings we have developed a number of open source tools that support transformations between some of the common formats encountered when developing an ontology-driven Web application. The Semantic Web Rule Language (SWRL) is a central building block in these tools. We describe these tools and illustrate their use in the development of a prototype Web-based application.

Introduction

Web-based applications typically deal with a variety of information formats. In addition to HTML and its variants, relational databases and XML-described information sources are common. Developing ontology-driven applications presents the additional challenge of dealing with yet another format, i.e., the system or domain ontologies used in the application itself. These ontologies should—in principle at least—represent the core format of knowledge in an application so all other formats used in a system may have to be mapped to integrate with them. The types of mapping required range from low level structural or syntactic mappings to transformations that require extensive domain knowledge.

We have identified four common types of transformations or mappings that are required when developing a Web-based application. These mappings ultimately involve describing the various formats in terms of the domain or system ontologies driving the application. These mappings are:

Relational Mapping. Relational databases will continue to be used in web applications for the foreseeable future, particularly if scalability is an important goal. Most web applications use a relational back end to store system data, which is usually accessed through an ODBC bridge or an object-relational layer. To deal with these data in ontology-based applications, efficient run-time mappings are required to deal with these data at the knowledge level.

XML Mapping. Information exchange in web-based applications is often described using XML. Indeed, these custom XML formats are now central to information exchange in many web-based applications and are used extensively when integrating with legacy systems. Extracting information from XML streams and describing it in terms of ontology–defined concepts requires a mapping process.

Ontology Mapping. Many ontology-driven applications must deal with multiple ontologies so a mapping process to integrate the information contained in these ontologies is usually required.

Application Data Mapping. The internal data structures of application software may also be considered an additional information format. The extraction of ontology information into these structures also constitutes a mapping process. This process is analogous to interacting with relational databases in general and requires equivalent tool and language support for ontology querying.

We have developed a number of open source tools using Semantic Web technologies to perform these mappings. These tools use the Web Ontology Language (OWL; [OWL, 2004]) and are built using the Protégé-OWL [Knublauch et al., 2004] ontology development platform. A central driving component of these tools is the Semantic Web Rule Language (SWRL; [Horrocks et al., 2004]). SWRL was developed as a formal description logic-based extension to OWL and provides an expressive language that is strongly suited to types of transformations required when performing knowledge mapping.

We first give some background on Semantic Web rule development with SWRL and describe an open source environment that we have written to work with SWRL rules. We then describe the tools we have produced to perform the four types of mappings outlined in this paper. Finally, we describe how we have used these tools in the development of a prototype Web-based application.

Background and Previous Work

The Semantic Web project is a shared research plan that aims to provide explicit semantic meaning to data and knowledge on the World Wide Web [Lee et al., 2001]. Semantic Web applications aim to be able to integrate data and knowledge automatically through the use of standardized languages that describes the content of Web-accessible resources.

OWL and SWRL

OWL and SWRL are core Semantic Web languages. OWL was developed as an ontology language for constructing ontologies that provide high-level descriptions of Web content. These ontologies are created by building hierarchies of classes describing concepts in a domain and relating the classes to each other using properties. OWL can also represent data as instances of OWL classes—referred to as individuals—and it provides mechanisms for reasoning with the data and manipulating it. OWL also provides a powerful axiom language for precisely defining how to interpret concepts in an ontology.

Recent work has concentrated on adding rules to OWL to provide an additional layer of expressivity. SWRL is one of the results of these activities. SWRL allows users to write rules that can be expressed in terms of OWL concepts and that can reason about OWL individuals. One of SWRL's most powerful features is its ability to support built-ins [Horrocks et al., 2004]. Built-ins are user-defined predicates that can be used in SWRL rules. A number of core built-ins for common mathematical and string operations are defined in the SWRL proposal. SWRL allows new libraries of built-ins to be defined and used in rules. Users can define built-in libraries to perform a wide range of tasks. Such tasks could, for example, include currency conversion, temporal manipulations, and taxonomy searches. In general, the arguments to these built-ins should be OWL DL property values—that is, literals or individuals. However, class or property built-in arguments may also be supported by some built-in libraries, though such built-ins should only be used in OWL Full ontologies.

SWRLAPI

We have developed several open-source tools to work with SWRL. One of the primary results is the SWRLAPI[1], an extension to the widely used Protégé-OWL ontology development toolkit [Knublauch et al., 2004]. The SWRLAPI provides a set of APIs that support the building of tools that work with SWRL rules. It has several software components, including: (1) an editor that supports interactive creating, editing, reading, and writing of SWRL rules; (2) a rule engine bridge that provides the infrastructure necessary to interoperate with third-party rule engines and reasoners; (3) a built-in bridge that provides a mechanism for defining Java implementations of SWRL built-ins; and (4) a variety of built-in libraries.

SWRL Editor. The Protégé-OWL SWRL Editor is an extension to Protégé-OWL that permits editing of SWRL rules. Users can interactively create, edit, and read/write SWRL rules. It is tightly integrated with Protégé-OWL and is primarily accessible through a tab within it. When editing rules, users can directly refer to OWL classes, properties, and individuals within an OWL ontology. They also have access to all the built-in libraries provided by the SWRLAPI.

Rule Engine Bridge. The SWRL Rule Engine Bridge is a subcomponent of the SWRLTab that provides a bridge between an OWL model with SWRL rules and a third party rule engine or reasoner. Its goal is to provide the infrastructure necessary to incorporate rule engines and reasoners into Protégé-OWL to execute SWRL rules. A bridge to the Jess rule engine is provided together with a user interface component [Friedman Hill, 2003]. The Pellet reasoner is also supported [Sirin et al, 2005].

Built-In Bridge. The SWRLAPI's Built-in Bridge[2] provides support for defining built-in implementations written in Java and dynamically loading them. Users wishing to provide implementations for a library of built-in methods can define a Java class that contains definitions for all the built-ins in their library. The bridge has a dynamic loading mechanism to import these built-in definitions and provides an invocation mechanism to execute these loaded definitions from rule engines. If additional rule engines are integrated into the SWRLAPI they can use these existing built-in libraries without modifying them.

Built-In Libraries. Using the built-in bridge we have developed a set of libraries for common methods required by rules[3]. These include implementations for the core SWRL built-ins defined by the SWRL Submission, a mathematical library that supports the use of complex expressions in rules, a temporal library that supports reasoning with temporal information, a library for dealing with XML documents, and libraries with ontology TBox and ABox operators. Libraries for dealing with spreadsheet documents, WSDL, SOAP, and RSS feeds are under development.

[1] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLAPI

[2] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLBuiltInBridge

[3] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries

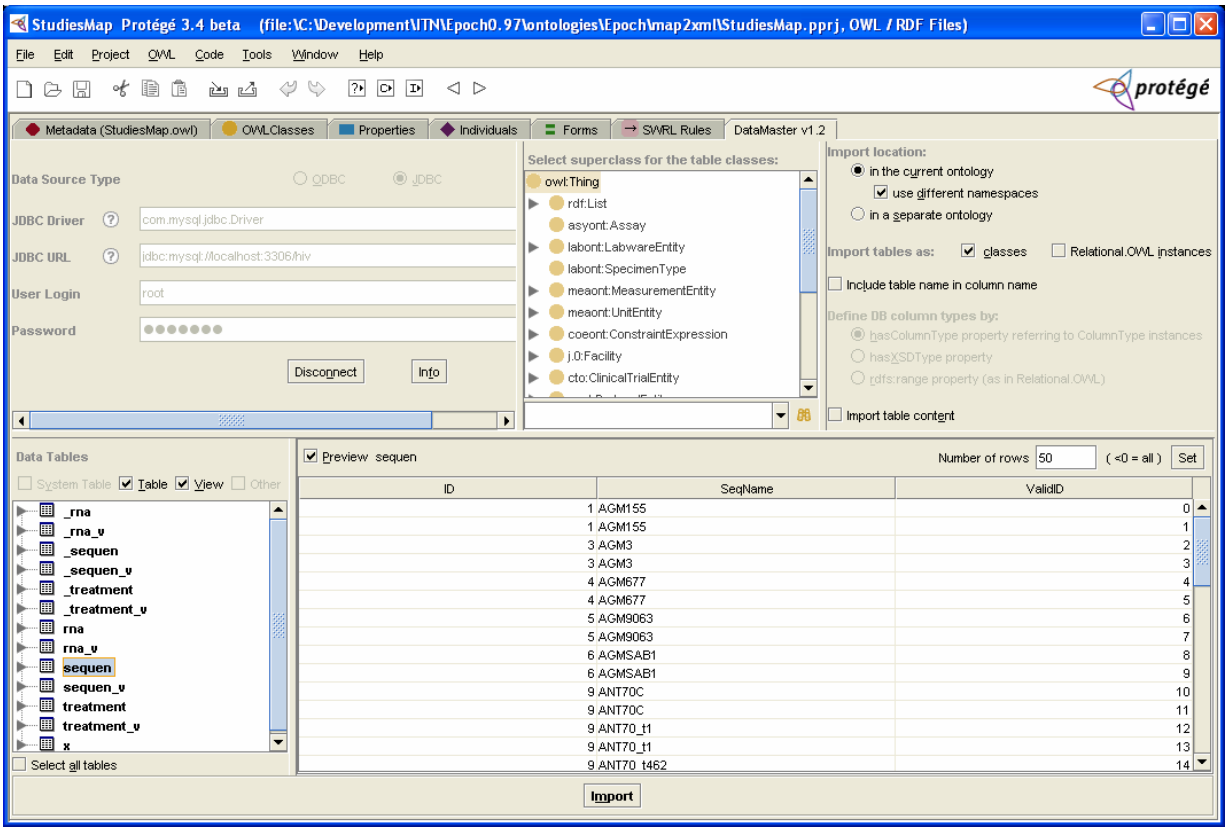

Figure 1. Screenshot of the DataMaster tab in Protégé-OWL. Users can connect to a relational database through ODBC or JDBC and import data into an OWL ontology.

Mapping Tools

Using the SWRLAPI, we developed tools to support the four types of mappings described in this paper. These tools are: (1) DataMaster[4] and Dynamic DataMaster, tools that supports mapping of relational data to OWL; (2) XMLMapper[5], a library that supports the mapping of information described in XML documents to and from OWL ontologies; (3) a set of SWRL built-in libraries that support ontology mapping; and (4) SQWRL, a SWRL-based query language that supports extraction of knowledge from OWL ontologies. SWRL's built-in mechanism provided the extension point for dealing with these additional information formats.

DataMaster

We have developed a relational-to-OWL mapping tool called DataMaster to serve as a relational data access mechanism in ontology-driven applications (Figure 1). It supports the mapping of data stored in relational databases to concepts described in OWL. DataMaster has two primary components: (1) schema and mapping ontologies that describe both the schema of an arbitrary relational database and the mapping of data stored in these schemas to triples in an OWL ontology; and (2) software to produce these ontologies, either automatically or semi-automatically with user guidance.

The OWL schema ontology provides a knowledge-level description of a relational schema. It describes schemas in a database and associated tables together with the columns and column data types contained in each table. It also describes primary and foreign key relationships for tables in a schema. We have also developed a mapping ontology that uses this schema ontology to describe how relational tables are to be mapped to OWL concepts. The fundamental goal is to specify the mapping of rows in a relational table to triples in an RDF model, which will then be mapped to OWL classes, properties and individuals. This process cannot normally be performed automatically, and additional user markup is usually required.

DataMaster can be used with any relational database with JDBC/ODBC drivers. A user interface is provided that supports user-driven configuration of the importation process. A user can use this interface to connect to a database and select the portions of the database that they wish to import. If the user decides on a schema-only import the database schema is read and represented in

[4] http://protegewiki.stanford.edu/index.php/DataMaster

[5] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabXMLBuiltIns

OWL using the Relational-OWL [de Laborda et al., 2005] ontology. If a content import is requested, the user can select a number of mapping options, such as, for example, how table and column names are mapped, and how relational column types are mapped to XSD Schema types.

Dynamic DataMaster

DataMaster also has a dynamic mapping layer that uses the schema source and mapping ontologies to map data on demand [O'Connor et al., 2007]. Instead of importing all database content, it provides a data access layer to dynamically translate knowledge-driven data requests to queries on a relational database. This layer works with an ontology level query engine to allow retrieval from a relational database. At run-time, the software uses the source and mapping ontologies to transform the data in a relational database to OWL entities. We extended our existing query engine to interact with this software to retrieve these mapped OWL entities. The engine takes queries written in terms of OWL classes, properties, and individuals and generates requests to the mapping software for the OWL entities in a relational database. The mapping software then generates SQL queries to retrieve that appropriate data from the database identified by the schema and mapping ontologies.

XMLMapper

We have developed a library called XMLMapper to read and write XML streams. This library can be used to read an XML document and automatically transform it to an OWL ontology that represents the document. This OWL representation is described in terms on an OWL XML ontology. This ontology has classes that represent standard XML constructs, such as document, element, attributes, and namespaces. Instances of this ontology are used to represent both the structure and content of an imported XML document. The reverse transformation from this ontology to XML is also supported.

Mapping Libraries

Information imported into OWL from other formats is generally not in a form that is directly usable with the domain ontologies used in an application. Typically, the imported information must be transformed or mapped to terms or concepts used in these domain ontologies. Similarly, third party ontologies that are used in the application will often have different representations of similar concepts. Or such ontologies may not be represented in OWL and may use RDF or other formats. In general, custom rule sets are needed to perform mappings because it requires domain-specific knowledge.

This mapping can be performed using SWRL rules. These rules take instance data represented using the source formats as imported into OWL and generates new instances that are described in terms of the local domain ontologies in the application. Irrespective of the source formats, there will be a common set of mapping operations that are required to reconcile these imported formats with application ontologies. When exporting knowledge from domain ontologies, a corresponding set of reverse mappings will be required. We have developed a suite of SWRL built-in libraries[6] that contain common operations that are typically required when performing these mapping. These operations support operations that are common in: (1) XML to OWL and OWL to XML mapping; (2) relational database mappings; (3) RDF mappings; (4) and OWL to OWL mappings.

Supporting OWL Queries with SQWRL

SWRL is a rule language, not a query language. However, many ontology-based applications require the ability to extract information from ontologies in addition to reasoning with the information in those ontologies. To support this knowledge extraction, we have developed a query language called SQWRL[7] (**S**emantic **Q**uery-Enhanced **W**eb **R**ule **L**anguage) that extends SWRL to support querying of OWL ontologies.

SQWRL is implemented as a built-in library using the standard SWRL built-in mechanism. It is syntactically and semantically compatible with standard SWRL. The SQWRL built-in library contains SQL-influenced built-ins that can be used in a rule to construct retrieval specifications for information stored in an OWL ontology.

For example, the following SQWRL query retrieves all persons in an ontology whose age is less than 25, together with their ages:

```
Person(?p) ^ hasAge(?p,?a) ^
swrlb:lessThan(?a,25) -> sqwrl:select(?p,?a)
```

This query will return pairs of persons and ages. The `sqwrl` prefix is used to identify SQWRL operators.

To list all cars owned by each person, we can write:

```
Person(?p) ^ hasCar(?p,?car) ->
sqwrl:select(?p,?car)
```

This query will return pairs of individuals and their cars. Assuming a person can have more than one car, multiple pairs would be displayed for each person. Basic counting is also supported by SQWRL, provided by an operator called `count`. If we wished to, for example, get a count of the number of cars owned by individuals in an ontology, we could write:

```
Person(?p) ^ hasCar(?p, ?c) -> sqwrl:select(?p) ^
sqwrl:count(?c)
```

This query would return a list of individuals and counts, with one row for each individual together with a count of the number of cars that they own. Individuals that have no cars would not be matched by this query.

---

[6] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTabBuiltInLibraries

[7] http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL

Basic aggregation is also supported. Four operators called `min`, `max`, `sum`, and `avg` provide this functionality. For example, a query to return the average age of persons in an ontology (for which an age is known) can be written:

```
Person(?p) ^ hasAge(?p, ?age) -> sqwrl:avg(?age)
```

SQWRL has access to all available SWRL built-in libraries. For example, to retrieve the name of all males in an ontology and prepend the title "Mr." to each name, we can use core SWRL `stringConcat` built-in:

```
Person(?p) ^ Male(?p) ^ hasName(?p, ?name) ^
swrlb:stringConcat(?fullname, "Mr. ", ?name) ->
sqwrl:select(?fullname)
```

The ability to freely use built-ins in a query provides a means of continuously expanding the power of the query language. Crucially, it provides an interoperation bridge to deal with other knowledge formats. SQWRL itself is formally based on OWL DL and SQWRL queries can be expressed only in terms of core OWL concepts. However, the built-in mechanism provides the ability to refer to non OWL DL concepts in queries. The TBox built-in library, for example, allows direct querying of OWL classes and properties, something that would not be possible in OWL DL. Similarly, we have developed an RDF built-in library that supports querying of RDF ontologies.

The SWRLAPI provides a graphical interface called the SQWRLQueryTab to execute SQWRL queries. A query can be selected from the rule table in the Protege-OWL SWRL Editor and executed to display query results. Users can navigate to that sub-tab to review the results displayed in tabular form. A JDBC-like Java interface is also provided to execute SQWRL queries in Java applications. Results for a particular query can be retrieved from a SWRLAPI rule engine bridge. Rows in a result can be iterated through and the contents of each column in a row retrieved using accessor methods. In addition to standard XML schema datatypes, OWL entities such as classes, properties, individuals, class descriptions and axioms can also be returned from queries.

Developing a Web Application

Using these mapping tools we developed a prototype Web-based application to advise physicians on HIV drug therapies. This application aims to replicate some of the functionality of the HIVdb application [Rhee et al., 2006], a Web-based utility that allows physicians to explore treatment options for HIV positive patients. The HIVdb application allows physicians to enter mutation and treatment histories for a patient and then presents them with treatment recommendations based on its analysis of patients with similar profiles stored in its database. The application is backed by an extensive curated research database that contains time-stamped data on drug regimens, HIV reverse transcriptase and protease sequences, and HIV viral load collected at local clinics.

To drive our prototype, we developed a central domain ontology called the *HIV Ontology* to represents all core concepts in the application. This ontology includes a *Virtual Patient Record* (VPR) ontology that holds clinical

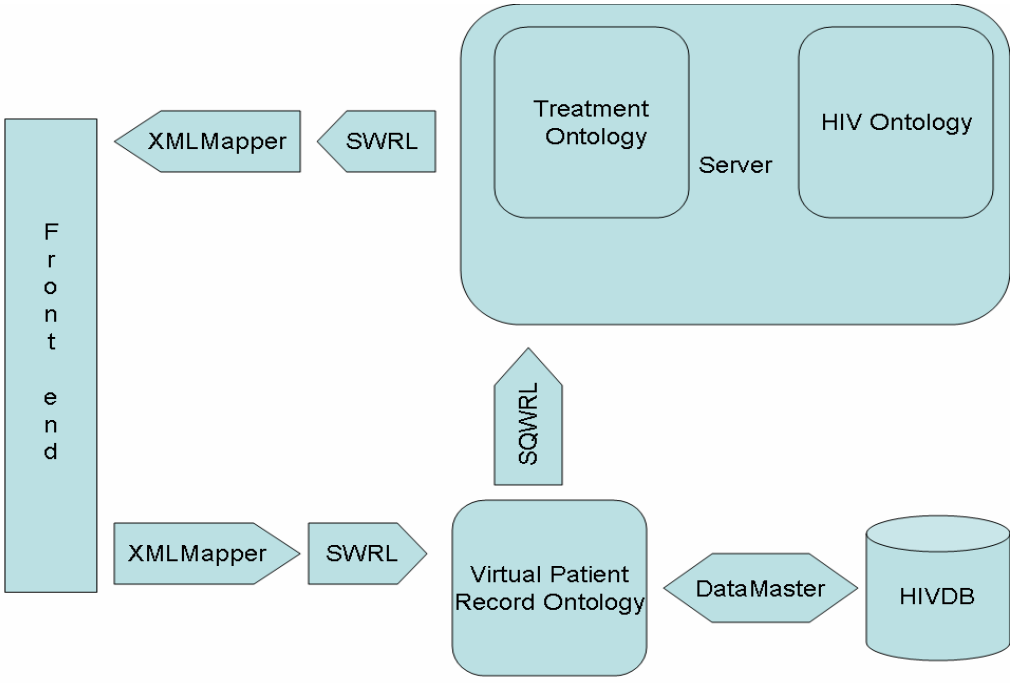

Figure 2. Component view of the Web-based application. Arrow-shaped components represent the various transformations or mappings that are performed in the system.

information for each patient, and a *Treatment Ontology* that contains therapy recommendations.

Figure 2 shows a high level component view of the application. We wrote a Web-based forms-driven front end that allows users to enter patient information. This information includes mutation and drug treatment history for a single patient. When the user indicates that information entry is complete, the information is transmitted to the server as a custom XML stream. At the server we use the XMLMapper library to transform this XML stream to OWL entities representing the content on that stream. A set of SWRL mapping rules is then executed to transform this information to instances described in terms of the HIV Ontology.

We used Dynamic DataMaster to describe mappings between the VPR and information in the HIVdb database. Based on the patient profile entered the application tries to find other patients in the database with similar profiles. It extracts this information using SQWRL queries written in terms of the VPR ontology, which are automatically mapped by Dynamic DataMaster to queries on the underlying relational database. The application then analyses the retrieved information and determines treatment recommendations for the patient, which are then stored in the Treatment Ontology. A second set of SWRL mapping rules is then activated to transform these recommendations to instances of the OWL XML ontology, which are mapped to an XML stream using the XMLMapper. This stream is then returned to the front end application, where it is parsed and summarized.

Conclusions

Data integration is a core central challenge of the Semantic Web. The ability to meet this challenge requires the development of a variety of mapping technologies to allow interoperation between the various formats that will be encountered when developing Semantic Web applications. The tools outlined in this paper provide a set of basic building blocks that can be used to construct these mapping technologies. We have found that SWRL can act as a central driving mechanism in these tools.

In addition to the prototype web application we have described, we have used these mapping tools to help meet the data mapping requirements of several ontology-driven biomedical applications [Shankar et al., 2008, O'Connor et al., 2008]. SWRL is used to help unify the domain-level specification of system data with the mapping needs of system components. In conjunction with OWL, SWRL provides both a formal domain-level description of data in the systems that we are developing and mapping mechanisms to transform these data between the various formats. These software components are open source and available in the standard Protégé-OWL distribution.

Acknowledgements

This work was supported in part by the Immune Tolerance Network, which is funded by the National Institutes of Health under Grant NO1-AI-15416, and also by the Centers for Disease Control and Prevention under grant number SPO-34603. We thank Valerie Natale for her editorial comments

References

Berners-Lee T, Lassila O, and Hendler J. *The Semantic Web*. Scientfic American, May 2001.

Freidman-Hill E. *Jess in Action.* Manning, 2003.

Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C, May 21, 2004.

Knublauch H, Fergerson RW, Noy NF, and Musen MA. *The Protégé-OWL Plugin: an open development environment for Semantic Web applications*. 3rd International Semantic Web Conference, Japan, 2004.

de Laborda CP, Conrad S. *RelationalOWL - a data and schema representation format based on OWL*. Conceptual Modelling, 43:89-96, 2005.

O'Connor MJ, Shankar RD, Tu SW, Nyulas C, Parrish DB, Musen, MA, Das AK. *Using Semantic Web Technologies for Knowledge-Driven Querying of Biomedical Data*. 11th Conference on Artificial Intelligence in Medicine (AIME 07), Amsterdam, Netherlands, 2007.

O'Connor MJ, Shankar, RD, Parrish DB, and Das AK. *Knowledge-Level Querying of Temporal Patterns in Clinical Research Systems*. International Journal of Medical Informatics, *in press*, 2008.

OWL Web Ontology Language Reference. www.w3.org/TR/owl-ref, 2004.

Rhee SY, Fessel WJ, Zolopa AR, et al., 2005. *HIV-1 protease and reverse-transcriptase mutations: Correlations with antiretroviral therapy in subtype B isolates and implications for drug-resistance surveillance.* Journal of Infectious Diseases 192: 456-465.

Shankar RD, Martins SB, O'Connor MJ, and Das AK. *An ontological approach to representing and reasoning with temporal constraints in clinical trial protocols.* International Conference on Health Informatics, Madeira, Portugal, 2008.

Sirin E, Parsia E, Grau BC, Kalyanpur A, and Katz Y. *Pellet: A practical OWL-DL reasoner*. UMIACS Technical Report, 2005-68, 2005.