UniProt in RDF Format: Use Cases and Open Issues

By Eric Jain, Swiss Institute of Bioinformatics

August 18, 2004

UniProt [http://uniprot.org/] is a comprehensive repository of protein sequence and annotation data. We have recently created an experimental version of the data in RDF format and developed several applications that demonstrate the usefulness of this approach [http://www.isb-sib.ch/~ejain/rdf/]. Following is an overview of use cases and open issues that we consider to be important.

Use Cases

1. Build tools that work with data from different data sets

Some tools are necessarily specific to a single data set. Most code however is concerned with moving data between different representations such as relational and file system storage. By using a generic data model this code does not need to be rewritten or adapted for each data set.

2. Merge data from different data sets

The boundaries between different data sets are more often than not dictated by practical rather than logical considerations. Unless data sets are logically or physically merged building queries that span several data sets requires detailed knowledge of the physical layout of the different data sets that are being queried. But writing code to merge two data sets can be time consuming unless both data sets use the same data model.

3. Change data layout without breaking existing tools

Minor changes to the way we represent data are introduced nearly every month; major changes several times per year. Some changes will inevitably cause some applications to break, but no application that does not directly rely on a piece of data that has been changed should break.

4. Detect conflicts and missing data

Data validation is usually too complex a task to be enforced completely by the representation of the data. The task of loading data into a rule engine is simplified a lot with a generic data model. A schema language helps define basic rules.

5. Allow third party annotations

A generic linking mechanism allows users to attach their own information to our data without requiring any special software to be set up or running into any intellectual property issues.

6. Track provenance of data

We need to indicate what data is supported by experimental evidence, and what data is based on specific prediction programs and models. Reification provides a logically correct way to represent this kind of information.

Open Issues

1.

The XML serialization syntax is confusing and difficult to process because it is very flexible. A possible solution is to restrict the syntax to a subset.

2.

The representation of ordered values is awkward. While most tools support collections and lists, few query engines support natural queries on such constructs.

3.

Direct support for reification is often neglected: OWL does not allow definition of complete subclasses of rdf:Statement. Neither BRQL nor N3 contain any constructs for convenient expression of reification. Most parsers return triples rather than quads, which complicates handling and reduces efficiency when working with data that contains a lot of reified statements.

4.

There is no support for negation. It should be possible to assert that a statement is not present. For example, a biologist may decide that a sequence feature prediction program generated a false positive hit. We would now like to add a statement that will cause a conflict if someone later on asserts that the feature is present.

5.

There is no reliable way to split a URI into a namespace and name part. While RDF itself is only concerned about complete URIs, many tools require such processing for simplicity and efficiency.

6.

Creating web services that return RDF data is complicated and inefficient. Arguably the same could be said of web services in general...

7.

The LSID [http://www.i3c.org/wgr/ta/resources/lsid/docs/] resolver mechanism is too complex and not suitable for batch lookups. A possible solution would be to decouple resolution from data retrieval and use a lightweight mechanism based on HTTP redirection, similar to the way DOIs [http://doi.org/] are resolved.

8.

There doesn't seem to be a mechanism for defining inline resource references. While we try to minimize such occurrences we are not able to avoid them completely. Example: "The authors of {#paper-1} disagree with {#paper-2} on..." The same problem can also be seen in the RDF version of the data provided by the Cyc project [http://www.cyc.com/2004/06/04/cyc].

9.

The XML serialization format assumes that related groups of statements can be managed by storing them into separate files. Therefore no mechanism for grouping statements within a file is provided. This is unfortunate, as this approach is not feasible when using files to exchange large data sets.

10.

There are currently no RDF editors available that are suitable for rapid data entry. Protégé is an excellent tool for defining classes and properties, but not efficient enough for fast data entry as too much clicking is required.

11.

Currently very few life science resources have stable URIs. This complicates linking. Resolver services could be set up as a temporary workaround.