

Experience Using OWL DL for the Exchange of Biological Pathway Information

Alan Ruttenberg¹, Jonathan A. Rees², Joanne S. Luciano³

¹ Millennium Pharmaceuticals, Inc., Cambridge,
Massachusetts 02139 USA
alanr@pathways.mumble.net

² CSAIL, Massachusetts Institute of Technology, Cambridge,
Massachusetts 02139 USA
jar@csail.mit.edu

³ Department of Genetics, Harvard Medical School Boston,
Massachusetts 02214 USA
jluciano@genetics.med.harvard.edu

Abstract. We report on experiences using OWL DL in the design of an exchange format for biological pathway information. Although the working group charged with this task was not initially very familiar with OWL and knew that the technology around OWL wasn't mature, they chose it because of its ability to express complex relationships in a formal and computable manner. The subsequent journey has not been smooth. Delightful discoveries about OWL have alternated with surprises about how difficult it is to operate correctly inside open world description logics and the Semantic Web generally. This paper highlights experience that may be of interest to the OWL community, including ontology developers, tool developers, and those interested in promoting the adoption of the Semantic Web.

1 Introduction

In 2001 the biomolecular pathways research community rallied around the idea of creating an open pathways resource akin to GenBank [1], the hugely successful community resource for genetics. The resource would collect pathway information, that is, information about interactions among biological entities and their effects on larger biological phenomena. Such a resource would require a common format for representation and transmission of pathway information, so a working group formed to develop such a format. The initial working group consisted mostly of representatives of diverse and already mature data curation and compilation efforts: BioCyc [2], WIT [3] (now Puma2), and BIND [4]. Later it grew to include several more pathway data sources, parties interested in biological knowledge representation, and users and integrators of pathway information [5].

For background, and to illustrate various issues involving OWL [6] use and adoption, we will discuss some activities of the working group. However, the views expressed in this paper are those of the authors and do not necessarily represent those of the working group.

Several design criteria guide the development of the exchange format. It is to be a machine computable formal representation to enhance the utility of the data and enable reasoning. It should interface with existing standards to enable interoperability. It should be extensible in order to have the capacity to evolve with scientific knowledge. It should support expressive new curation adequate to represent the pathway knowledge expressed in scientific papers. Finally, because each participating source of pathway information represents its descriptions using its own semantics and data format, the common format should be suitable as a translation target for existing data.

Few in the working group had any prior stake in RDF [7], OWL, description logics, standards projects, or the Semantic Web. The focus was on exchanging application-level information; for many, work on the exchange format was a necessary evil.

The group was not initially very familiar with OWL, but after a one-day tutorial [8] it was sufficiently impressed by OWL's merits to take it seriously as a specification vehicle. OWL's ability to express complex relationships and constraints was judged a match to the group's goals, and they chose OWL DL (over XML Schema) as its ontology framework. The decision was not without controversy; XML Schema was favored by some because of its already wide adoption and abundance of tools. In the end OWL won because of its expressiveness and the expectation that if adopted by the W3C [9] tools and wide acceptance would follow.

The group initially used OWL as if it were like any of the other schema definition tools such as relational databases and XML Schema. There were certain expectations taken from these tools, such as the closed world assumption. In particular, they expected to simply invent a new, federated schema that unified common elements of the schemas of the existing data sources and was similar in kind to the schemas of the existing data sources. Some use of OWL's new features was expected, even though it wasn't clear how, when, or whether to use them. But no one was considering radical change relative to the way the existing schemas had been built, such as mapping data records to classes instead of instances.

In this paper we document a variety of issues that we hope contribute to the ongoing discussion of the use of ontologies in the context of the Semantic Web.

2 Ambushed by the open world assumption

The open world assumption says that anything not known to be true or false might become so as a result of new information. There are positive and negative aspects of OWL's open world assumption with respect to the stated design criteria. On the positive side, the open world assumption seems particularly fitting in a domain that is characterized by information that is incomplete either because of limits in the state of knowledge or omissions inherent in curation processes. One can imagine a scenario in which this partial knowledge is augmented by subsequent contributors, in line with the goals of the Semantic Web.

On the negative side, the open world assumption has generated problems that were not anticipated:

No way to require that information be supplied. Sometimes information that is to be exchanged cannot, by its nature, be reconstructed or added to. Consider a reference

to a paper. Currently this is represented by a pair of string valued properties: database name and database identifier. One wants to say that that each of these properties needs to have values if one is to make any sense of the reference. OWL can express something like this using *minCardinality* constraints. However, if one of the properties doesn't have a value, no OWL validator will complain, since under the open world assumption, the property could be asserted later. But consider the task of annotating that an interaction between two proteins was noted in a particular journal article. If one says that the database is *PubMed* but doesn't fill in the article identifier, one cannot identify the article. What one wants is the ability to express that *within a given scope*, certain restrictions must be verifiable with the assertions expressed. That way one could express, for instance, that within the assertions in a single file, or at a single URL, any reference to a publication must have values for both the publication and identifier properties. Note that while this corresponds to "closing the world" over the specified scope, there is no requirement that it stay closed, nor that it affect the semantics of the document outside the scope.

No convenient way to assert that information is complete. On the other side of the open world assumption is the situation where we have a property whose value is completely known. For example, in the specification of an instance of some protein complexes, we want to assert that we have listed all the components of the complex. In order to do this we need to "close" the components role, making such a complex an instance of a restriction of a cardinality constraint on the components¹. In Protégé, for instance, there is no convenient way to assert such a constraint.

Unique name assumption difficult to understand and maintain. Removing the unique name assumption is a useful idea on the greater Semantic Web, where it is likely that people can name the same concept in different ways. However, within a single source of information we generally know that different names name different objects. It is inconvenient to maintain all the *differentFrom* assertions as a document evolves. It is also tricky to assess the consequences of getting this wrong. This is another case where the concept of scope might be useful, specifically the ability to assert that all names within a scope represent different things.

Novices are confused about properties that are not asserted. For example, in the description of a chemical reaction there is a property for stoichiometry (multiplicity of a reactant). Since the most common case is that stoichiometry is 1, it was suggested that in order to make the documents less verbose that an unasserted stoichiometry would be taken to mean 1. However, in OWL an unasserted property means that the value is unknown. This was a surprise to most of the group. While we would like to propose some technical fix for this, we can't think of one.

¹ Closing a role by adding a property restriction type to an instance:
`Individual(Instance type(complex) type(restriction(component cardinality(2))))`

3 Using other ontologies

There has been substantial prior work on developing ontologies relevant to representing pathway information and the exchange format would like to be able to take advantage of this work. For example, post-translational modifications are described in RESID [10] and portions of PSI MI [11], while cellular locations are described in portions of the Gene Ontology (GO) [12]. How these external entities to be used in the ontology?

Few of these ontologies are provided as OWL DL. Currently, terms from such ontologies are represented as values of two properties, one giving a name for the vocabulary from which the term was taken, and the other identifying the term in the vocabulary. Unfortunately for the Semantic Web, neither the terms nor the names of the vocabularies are URIs. Moreover, external terms are not just meaningless data; some understanding of them is required for reasoning and validation. Representing terms in this way, semantic relations, such as the containment relationship between cellular locations, are lost. Some properties should be restricted to particular classes in GO; a property denoting a cellular location cannot be filled with a term which is a subclass of *molecular function*.

An alternative approach would be to first create OWL versions of the needed ontologies and then import them, thereby making all available information directly accessible. As an experiment one of the authors (AR) wrote translators to convert the relevant portions of PSI MI to OWL DL. We first identified a portion of the vocabulary that would be used to annotate post-translational modifications, namely the terms in the hierarchy below *MI:0120* other than *MI:0179*. The *is_a* relationships were translated into subclass relations in OWL. Annotation properties were used to record additional information about the terms, such as synonyms, English definitions, and identifiers.

Another question is the treatment of changes to the external ontology. If we choose to have references to terms in the external ontology, we may be left with incorrect identifiers in our documents when terms in the external ontology are deleted or deprecated. This is particularly an issue with the rapidly changing Gene Ontology. However, if we translate and import an external ontology, new and changed terms will not be available for use until we update our translation. On the other hand, a user of our ontology will benefit from the stability of knowing the potential term set in advance.

4 Getting validation

The typical software engineer expects a rapid edit-compile-debug development cycle, and on starting to work with OWL, one expects to be able to iterate in a similar manner. In place of compilation one would like to check that the file is formatted correctly, that the definitions make sense, and that the inferences one expects to make can in fact be made. Unfortunately one is immediately hindered by the inability to reliably do so.

Checking that the file is formatted correctly and that the definitions make sense is the role of a validator. One expects a validator to assess whether the file complies

with the specification and to generate specific detailed reports when it doesn't. Not having such a tool makes it difficult for data providers to check whether their code is generating correct OWL.

Checking that the inferences one expects to make can in fact be made is the function of a reasoner. For OWL DL we expect that a reasoner is able to test whether the ontology (including both classes and instances) is consistent, to respond to queries asking for equivalences, superclasses and subclasses of a given class, what instances are members of a class, what the classes of an instance are, and what the values of properties are. Not having a reasoner makes it difficult for the novice ontologist to check whether they understand the implications of their modeling choices. Without a reasoner one cannot build clients of the exchange format that can take advantage of the promised expressiveness of OWL. Since a good validator must make some use of a reasoner, lack of a reasoner hinders efforts to build a robust validator.

In trying to find reasoners and validators we first checked the OWL test site [13], which was not reassuring. Based on the test results presented there, it seems that a reasoner that is complete with respect to OWL DL does not yet exist.

We reviewed some of the available tools, using the most recent versions available when we did the evaluation in mid July, 2005: Protégé [14], SWOOP [15] with the Pellet [16] reasoner, Racer Pro [17] (both as a DIG [18] server for Protégé and as a standalone application), the BBN OWL Validator (vOWLidator) [19], and FaCT [20,21]. In response to reviewer's comments we also reviewed the OWL API [22], the WonderWeb OWL Ontology Validator [23] and the Pellet reasoner (standalone) using the versions available at the beginning of October, 2005. All these systems had issues.

First we explored validation and reasoning using Protégé. Protégé does some reasoning on its own and also provides an interface to external DIG reasoners. Protégé's native validation and reasoning support is spotty. It doesn't do subsumption reasoning. It does do some role reasoning, such as inferring the values of properties when sub-Property values are asserted, but it doesn't mark inferred values distinctly in the interface, and doesn't serialize them to the saved OWL file. We think this patchwork approach to reasoning support will be confusing to the general user.

Using external reasoners from Protege is unsatisfactory because the DIG protocol doesn't support some constructs available in OWL DL, so one gets many spurious warnings, leading one to question the completeness of the validation. In fact it isn't. Consider the following ontology:

```
DatatypeProperty(Property1 range(xsd:string))
Class (Class1 partial)
Class (Class2 partial Class1 restriction(Property1 minCardinality(1)))
Class (Class3 partial Class2 restriction(Property1 maxCardinality(0)))
```

When we check ontology consistency we get the message *Not able to convert datatype property cardinality restrictions to DIG (the language used to communicate with the reasoner). Ignoring this restriction and attempting to continue*. Because of this, Protégé is not able to detect that Class3 is inconsistent. We tested this both with FaCT++ and the Pellet reasoner in DIG mode in late September, 2005. The Pellet web form, which accepts OWL directly, correctly notes the inconsistency.

SWOOP was not particularly robust. Enabling the reasoner while working on an ontology with an inconsistency often caused application errors that could not be re-

covered from. The debugging alpha version that we used did supply us, in one case, with a chain of assertions that supposedly led to an inconsistency. However, it was difficult to follow the logic, and as the inconsistency was not noted by either Racer or FaCT, we assumed that it was spurious. More detail can be found on the BioPAX wiki [24].

The Pellet reasoner, used as a standalone tool, looks very promising. In a recent test we found it useful in validating and debugging a large set of instances (several megabytes), issuing informative comments describing problems. It is not without limitations. In the days before finishing this paper, we identified two issues. To the credit of the developers, these were promptly fixed. However we are still able to find examples which provoke incorrect behavior. The following example is incorrectly classified as OWL DL. It is OWL FULL because of the cardinality constraint on the transitive property *part_of*.

```
ObjectProperty(part_of Transitive domain(Class1) range(Class1))
Class(Class1 partial restriction(part_of cardinality(1)))
```

vOWLidator does not recognize *oneOf dataRange* restrictions, and so it generates many spurious complaints that need to be examined and filtered out in order to find useful warnings. It doesn't check certain RDF/XML requirements such as the need for a data type on a property value whenever the property's range is restricted to a certain data type. When errors or warnings are reported, the notes often refer to the internal identifiers of blank nodes, which makes it difficult to find the source of the error in the ontology.

Racer Pro seemed robust and reliable at detecting inconsistencies and errors in some large OWL files. However, whereas it was able to detect inconsistencies (even in a property data type), it didn't report anything more than that the file was inconsistent. This made it difficult to find the source of the error. As we were trying to check an 18MB file containing the pathway content from HumanCyc [25], this wasn't very useful. Finally, Racer Pro is a commercial product. While some free licenses are available, they come under terms that were not satisfied by all members of the group.

FaCT is listed as an OWL DL reasoner. We downloaded the open source Common Lisp implementation hoping to use that. However, it has no defined OWL support, nor were we able to find a publication that showed how to translate even OWL DL TBox (class) reasoning into the API used by FaCT. We used Wilbur [26] to read the OWL RDF and wrote code (probably buggy) that translated the OWL primitives into the FaCT API and did get some useful information from it – the detection of an unsatisfiable class caused by multiple inheritance from two disjoint classes. Since we had access to the source, we were able to turn on debugging switches to more easily identify the source of the problem. However, since FaCT only supports TBox reasoning, we were unable to use it to validate any of our pathway data which primarily consists of instances.

Ian Horrocks pointed us to FaCT++ [27] as the current incarnation of FaCT, suitable for OWL reasoning. However, FaCT++ is described as a reasoner for OWL Lite,

and the status of using it as a reasoner for OWL DL was not, until recently¹, clear to us. We tried a recent version of FaCT++ DIG server within Protégé. In this mode it suffers from the noted limitations of the OWL to DIG translation.

We were unsuccessful at retrieving inferred property values with the OWL API. For example, values of a *subProperty* are also considered to be values of the property they descend from. In the following definition there are two properties *Property1* and a *subProperty* of it: *Property2*. The single instance has a value for *Property2*. We expect that one value of *Property1* would have this value. As best we could tell, one uses the function *anOwlInstance.getDataPropertyValue()* to retrieve property values for an instance. However this function returned no value for the parent property *Property1*.

```
DatatypeProperty(Property1 domain(Class) range(xsd:int))
DatatypeProperty(Property2)
SubPropertyOf(Property2 Property1)
Class(Class partial)
Individual(Instance1 type(Class)
  value(Property2 "1"^^<http://www.w3.org/2001/XMLSchema#int>))
```

The WonderWeb OWL Ontology Validator also had trouble with the example that generates DIG warnings in Protégé, shown above. It also considers that ontology consistent. We speculate that it suffers the same DIG imposed limitation on cardinality constraints as noted above. However, unlike Protégé, no diagnostics are given. In addition, it does not detect missing *rdf:datatype* statements, or values with inconsistent data types. For example, the following ontology is incorrectly considered valid. In fact any value and any *rdf:datatype* for *Property1* is considered valid.

```
DatatypeProperty(Property1 domain(Class) range(xsd:int))
Class(Class partial)
Individual(Instance1 type(Class)
  value(Property1 "1.1"^^<http://www.w3.org/2001/XMLSchema#float>))
```

Some constraints not expressible in OWL. As it turns out, OWL DL can't express all the constraints we care about in our domain. [28,29] For example, in a chemical reaction, matter is conserved, so there is a constraint that the total mass of the reactants is the same as the total mass of the products. Since we can't express such a constraint in OWL DL, we plan to have a separate, domain specific validator to ensure that such constraints are satisfied. We expect that other projects would be in a similar situation. However, we want to check as much as possible in OWL so as to reduce the effort in creating this auxiliary validator and to make those constraints known to reasoners.

¹ Ian Horrocks (personal communication) notes that there have been recent advances in the algorithms for reasoning with OWL DL, that these have been implemented in the Pellet system, and that they will soon be available in FaCT++

5 What, in the world, do we mean?

The data to be encoded in RDF/OWL consists of records describing molecular entities (such as metabolites and proteins), molecular complexes, metabolic reactions, and signaling pathways. It seems reasonable for RDF individuals to play the role of records, with properties acting as record fields. The ontology then plays the role of schema since it specifies what kinds of data can be the various properties of an object, and which objects can link to what other kinds of objects.

Database designers don't generally spend much time thinking about denotation and truth, but RDF and OWL impose a sort of moral imperative to address these issues somehow. In our case, the content being represented is about the world – so the world should be a model of our logical system, in the sense of the OWL formal semantics [30]. The challenge is to figure out what the correspondence is. The fact that we are designing an exchange language makes this question that much more important. Consider the alternative. If the specification doesn't carefully define the mapping of classes and instances to biological phenomena, each provider of information would have its own mapping and it would fall to clients using more than one source to figure out how to relate terms from the various sources. This would defeat the purpose of creating an exchange format. Thus we need to define what, in the world, our classes and instances correspond to. Doing so was not something the working group had anticipated.

The issue was first raised when trying to understand what it meant to refer to a given physical entity instance in more than one reaction. On the one hand, there is a desire to reuse instances because they are rather large, including information such as synonyms, chemical structure and so on. On the other hand, there is the intuition that referring to the same instance means referring to the same thing in the world. Since one can reuse the same instance of *protein* to describe interactions that take place in different places, it can't mean the same physical protein. If "same instance" can't mean "same protein", what does it mean?

To answer this you need to know what a protein "is" and, by implication, what it takes for two proteins to be different. Sameness could mean a particular protein molecule situated in space and time; a quantity or "pool" of protein belonging to some unspecified compartment model; or an idealized single molecule participating in a collective scientific drama ("P53 has a role in ..."). Differentness might or might not hinge on genetic polymorphisms, mutations, or post-translational modifications.

To further complicate the issue, consider the task of representing a homodimer, i.e. a molecular complex consisting of two copies of the same molecule. In this case we can't have both molecules be the same instance – if we did that we would be asserting the same *component* property twice, which is the same as saying it once in RDF. It was proposed that in this case we could use stoichiometry to represent the multiplicity. Consider then the situation where one of the two proteins is modified in a reaction, e.g. phosphorylated. Now the initial single instance becomes two instances.

If instances don't represent single things, should they not be represented as classes? If we use classes instead of instances to represent reactants, OWL DL forces us to use classes to represent reactions and other higher order entities, since there is a limited repertoire of ways to relate classes to one another. We know that if we only use

classes then our expressiveness is diminished compared to using instances, since instances can form cyclic graphs, but classes can't. Do we need that expressiveness?

Absent further guidance we are having trouble deciding where to draw the line.

6 Conclusions and recommendations

OWL continues to be the basis of BioPAX working group's specification efforts. Ontologies, instance data, documentation, and further discussion may be found online at the group's wiki, <http://biopaxwiki.org/>. However, in spite of the group's experience in biological knowledge representation, bioinformatics, software engineering, and database design, it encountered some challenging problems. We think problems similar to those described above will be common as more groups try to interact with the Semantic Web.

Semantic Web technology is not yet mature. Users of OWL should expect to engage OWL tool developers, expecting and reporting bugs and challenging developers to address their needs. Designers and advocates of the Semantic Web need to get involved in more concrete projects to gain a better understanding of their audience.

In order to support efforts to refer to objects in RDF, database providers of all sorts should be encouraged to define URIs that identify their objects. Correspondingly the Semantic Web community should supply guidance on how to do so, and provide strategies for OWL users to cope with references to such entities when they are not supplied.

There will be a need for domain specific validation in many cases. In order to minimize the effort involved with implementing systems to do this, cases such as those we outline in our discussion of open world issues could be reviewed. Where appropriate, additions to the specification, such as a theory of scope and statements over a scope, could address common cases that are somewhere between the limits of what is currently expressible in OWL and checks that can only be done with knowledge of the specific domain.

We can't emphasize enough the need for a freely available, open source, complete and accurate OWL validator and reasoner as a tool for making progress in using and understanding OWL. Since so many of the concepts that it uses are new to a general audience, and the implications of using various features non-obvious, lack of such a validator makes progress by experimentation extremely difficult. While Pellet seems to be a strong candidate for filling that role, we found it difficult to identify it as such given a wide choice of candidate systems. More effort needs to be devoted to maintaining a solid test suite and providing up-to-date reports on the status of the various systems. Finally, simply detecting problems is not enough. More research needs to be done to identify strategies to explain the reasons for inconsistencies and errors in ways that users can understand so they can learn enough to fix them.

The kinds of relationships that need to be modeled in biology are varied and go beyond class/subclass relationships, [31,32] for example part/whole relationships and *derives-from*, used to describe development of organisms. There is a wide gap between the expressiveness (and tractability) of OWL DL and OWL Full. OWL would benefit from the elaboration of other levels between these two, particularly if such

levels could afford additional expressiveness needed for biological description while still having some guarantees around the abilities of reasoners.

We are excited that RDF and OWL are not neutral data representation formats, but modes of expression that have the character of assertion. However, this comes at a cost. Someone generating RDF triples should feel accountable both for the meaning and the truth of their assertions. But this can be hard work. Because of this, there is a temptation to avoid defining terms precisely. But we're playing a new game now: we are obligated to document what we mean.

This is a double-edged sword, of course. On the one hand, hardly anyone wants to work out the details of the model since it appears to be an unglamorous, low-yield bit of intellectual drudgery and a threat to progress. On the other hand, if we're successful at defining our terms in such a way that our triples become plausible assertions, then users of the information will benefit enormously - in particular, they will be able to federate data and do their own inference without having to reverse engineer the meaning of each data source. The high quality technical framework that OWL provides forces a good technical approach. In the words of W3C Semantic Web Activity Lead Eric Miller, "You do it right once, and everyone benefits."

7 Acknowledgments

Discussions with Matthias Samwalt, Imre Vastrik, Dan Corwin, Frank Schaerer and Stan Letovsky were helpful and illuminating. Thanks to Jeremy Zucker for reviewing the manuscript. BioPAX is the work of the BioPAX working group: Mirit Aladjem, Gary D. Bader, Erik Brauner, Michael P. Cary, Dan Corwin, Kam Dahlquist, Emek Demir, Peter D'Eustachio, Ken Fukuda, Frank Gibbons, Marc Gillespie, Robert Goldberg, Chris Hogue, Michael Hucka, Geeta Joshi-Tope, David Kane, Peter Karp, Christian Lemer, Joanne Luciano, Natalia Maltsev, Debbie Marks, Eric Neumann, Suzanne Paley, Elgar Pichler, John Pick, Jonathan Rees, Aviv Regev, Alan Ruttenberg, Andrey Rzhetsky, Chris Sander, Vincent Schachter, Imran Shah, Andrea Splendiani, Mustafa Syed, Edgar Wingender, Guanming Wu, Jeremy Zucker. (The working group is a dynamic community. We apologize if we have omitted a member from this list.) JSL gratefully acknowledges the Office of Biological and Environmental Research Genomics: GTL Program (grant #DE-FG02-04ER63931), George Church, Robert Stevens and the National Science Foundation (grant #IIS-0542041) for their generous support.

References

1. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A., Wheeler, D.L.: GenBank. *Nucleic Acids Res.* **28** (2000) 15-18

2. Krieger, C.J. et al. MetaCyc: A Multiorganism Database of Metabolic Pathways and Enzymes. *Nucleic Acids Res.* **32** (2004) D438–442
3. Overbeek, R. et al.: WIT: Integrated System for High-Throughput Genome Sequence Analysis and Metabolic Reconstruction. *Nucleic Acids Res.* **28** (2000) 123–125
4. Bader, G.D. et al.: BIND: the Biomolecular Interaction Network Database. *Nucleic Acids Res.* **31** (2003) 248–250
5. <http://biopaxwiki.org/>
6. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference, 10 February 2004. Available at <http://www.w3.org/TR/owl-ref>
7. Klyne, G., Carroll, J.J.: Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 10 February 2004. Available at <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210>
8. http://www.biopax.org/Docs/2003-02-20_OntologyTutorial.ppt
9. <http://w3.org>
10. Garavelli, J.S.: The RESID Database of Protein Modifications: 2003 developments *Nucleic Acids Res.* **31** (2003) 499-501
11. Hermjakob, H. et al.: The HUPO PSI's Molecular Interaction Format – A Community Standard for the Representation of Protein Interaction Data. *Nat. Biotechnol.* **22** (2004) 177–183
12. The Gene Ontology consortium: Gene ontology: Tool for the Unification of Biology. *Nat. Genet.* **25** (2000) 25–29
13. <http://www.w3.org/2003/08/owl-systems/test-results-out>
14. Noy, N. F., Sintek, M., Decker, S., Crubézy, M., Ferguson, R. W., Musen, M. A.: Creating Semantic Web Contents with Protégé-2000. *IEEE Intelligent Systems* **16** (2001) 60-71
15. Kalyanpur, A., Bijan Parsia, B., Hendler, J.: A Tool for Working with Web Ontologies. *International Journal on Semantic Web and Information Systems*, **1** (2005)
16. Sirin, E., Parsia, B.: Pellet: An OWL DL Reasoner. *Description Logics 2004, CEUR Workshop Proceedings* **104** (2004)
17. Haarslev, V., Möller, R.: RACER System Description, *Lecture Notes in Computer Science*, **2083** (2001) 701
18. Bechhofer, S.: The DIG Description Logic Interface: DIG/1.1, available at <http://dlweb.man.ac.uk/dig/2003/02/interface.pdf>
19. <http://projects.semwebcentral.org/projects/vowlidator/>
20. Horrocks, I.: Using an Expressive Description Logic: FaCT or Fiction? In: Cohn, A. G., Schubert, L., Shapiro, S. C. (eds): *Proc. of KR-98*. Morgan Kaufmann Publishers, San Francisco, California (1998) 636-647
21. FaCT implementation available at <http://www.cs.man.ac.uk/~horrocks/FaCT/>
22. <http://owl.man.ac.uk/api.shtml>
23. <http://phoebus.cs.man.ac.uk:9999/OWL/Validator>
24. http://biopaxwiki.org/cgi-bin/moin.cgi/Known_OWL_validation_issues_with_vowlidator
25. Romero, P., Wagg, J., Green, M.L., Kaiser, D., Krummenacker, M. Karp, P.D.: Computational Prediction of Human Metabolic Pathways from the Complete Human Genome, *Genome Biology* **6** (2004) R2 R2.1-17
26. Lassila, O.: Wilbur Semantic Web Toolkit for CLOS. Available at <http://wilbur-rdf.sourceforge.net/>
27. Tsarkov, D., Horrocks, I.: Fact++ implementation. Available at: <http://owl.man.ac.uk/factplusplus/>
28. http://biopaxwiki.org/cgi-bin/moin.cgi/To_OWL_or_not_to_OWL
29. http://biopaxwiki.org/cgi-bin/moin.cgi/best_practices

30. Patel-Schneider, P.F., Hayes, P., Horrocks, I. (ed.): OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, February 2004. Available at <http://www.w3.org/TR/owl-semantics/>
31. Smith B., Ceusters W., Klagges B., Kohler J., Kumar A., Lomax J., Mungall C.J., Neuhaus F., Rector A., Rosse C.: Relations in Biomedical Ontologies. *Genome Biology* **6** (2005) R46.
32. Smith B.: The Logic of Biological Classification and the Foundations of Biomedical Ontology. Available at: http://ontology.buffalo.edu/bio/logic_of_classes.pdf