Next Steps for OWL

Bernardo Cuenca Grau[1], Ian Horrocks[1], Bijan Parsia[1], Peter Patel-Schneider[2] and Ulrike Sattler[1]

[1] School of Computer Science,
University of Manchester, UK
[2] Bell Labs Research, NJ (USA)

Abstract. OWL 1.1 is a simple extension of the OWL DL species of the W3C OWL Web Ontology Language. OWL 1.1 has been designed to provide some interesting and useful expressive additions to OWL DL while retaining the desirable characteristics of OWL DL, including decidability and implementability.

1 Introduction

OWL is an ontology language, or rather a family of three ontology languages, developed by the World Wide Web Consortium (W3C) as part of its Semantic Web activity [16]. The development of OWL was motivated by the key role foreseen for ontologies in the Semantic Web (i.e., providing precisely defined and machine processable vocabularies that can be used in semantically meaningful annotations), and the recognition that existing web languages, such as RDF and RDF Schema, were not expressive enough for this task [10].

The standardisation of OWL has led to the development and adaption of a wide range of tools and services. These include reasoners such as FaCT++ [20], Racer [5] and Pellet [18], and editing tools such as Protégé [17], Swoop [12], Ontotrack [13].

Although OWL was initially designed for use in (the development of) the semantic web, it has rapidly become a de facto standard for ontology development in general, see, e.g. BioPax (http://www.biopax.org/). This is probably due to the ready availability of a wide range of OWL tools, and the greatly increased potential for sharing and reuse provided by the adoption of a standard. OWL ontologies are now under development and/or in use in areas as diverse as e-Science, medicine, biology, geography, astronomy, defence, and the automotive and aerospace industries. Although this represents a considerable success story for OWL, such widespread use of the language has also revealed deficiencies in the original design, and led to requirements for language extensions. These included increased expressivity with respect to properties, number restrictions, and data-values, and some form of meta modelling [14].

On studying these requirements, it became clear that several of them were addressed, at least in part, by recent developments in DL languages and reasoning techniques. This led to the idea to develop an incremental extension of

OWL, provisionally called OWL 1.1, that would exploit these recent developments in order to provide a more expressive language, but one which retained OWL's desirable computational properties (in particular decidability) and which would allow for the relatively easy extension of existing reasoning systems in order to provide support for the new language. The features to include in OWL 1.1 were agreed on a year ago at the first OWL: Experiences and Directions workshop (http://www.mindswap.org/2005/OWLWorkshop/). In this paper, we summarize the work that has been done since then in nailing down the details of OWL 1.1 with the goal of starting a standarisation process. The documents that describe the current status of OWL 1.1 are available online in (http://owl1-1.cs.manchester.ac.uk/) and open for discussion.

2 Overview

The initial design of the OWL was (understandably) quite conservative, and features that did not receive widespread support within the working group were excluded from the language. Features for which effective reasoning methods were not known (or expected to be shortly known) were also not included.

As mentioned above, the use of OWL, particularly the OWL DL species of OWL, has identified several important features, support for which would greatly increase the utility of the language. Some of these, such as *qualified number restrictions*, were already supported by DL systems when OWL was designed, but were excluded from the language. Others, such as *complex role inclusion axioms*, could now be supported (at least in part) as a result of recent advances in DL theory. OWL 1.1 is a simple extension to OWL DL that:

1. adds language features commonly requested by users of OWL DL;
2. is known to be decidable, and for which practical decision procedures have been designed; and
3. is likely to be implemented by the developers of OWL DL reasoners.

The user requirements that drove the extensions in OWL 1.1, and the language features that address them, fall into five distinct categories:

Syntactic sugar: Some commonly used representations are difficult and/or cumbersome to express in OWL. For example, it is very common to assert that a number of classes are pairwise disjoint (this is often the default for the direct subclasses of a common parent class). Although this can be expressed in OWL, it is necessary to assert each pairwise disjointness separately (or to employ some representational "tricks"), which is cumbersome when large numbers of classes are involved, and may also make it more difficult for reasoners to optimise the way they deal with such sets of disjoint classes.

Increased expressiveness in property constructs: Although OWL is relatively expressive, there are still many situations that are difficult or impossible

to represent using OWL. In particular, while OWL provides a wide range of constructors for building complex classes, relatively little can be said about properties. A very common requirement is to express the "propagation" of one property along another property [19], e.g., it may be useful to express that certain locative properties are transfered across certain part-whole properties. For example, the OWL 1.1 axiom SubPropertyOf(propertyChain(locatedIn partOf)locatedIn) states that, if something is located in a part of an object, it is also located in the object itself; thus, when using a medical ontology, a lesion located in a part of a body structure is recognised as being located in the body structure as a whole. This enables highly desirable inferences such as a fracture of the neck of the femur being inferred to be a kind of fracture of the femur, or an ulcer located in the gastric mucosa being inferred to be a kind of stomach ulcer.

Furthermore, in OWL 1.1 properties can be made disjoint, and one can state properties to be reflexive, irreflexive, symmetric, or antisymmetric. For example the property motherOf should be irreflexive, because nobody can be a mother of himself, and the properties childOf and spouseOf are naturally disjoint (DisjointProperties(motherOf, childOf), since x cannot be the mother and the child of y at the same time.

Increased datatype expressiveness: OWL provides very limited features for describing classes whose features include concrete values such as integers and strings. It is a common requirement, for example, to express value ranges (a Gale is a wind whose speed is in the range 34–40 knots), or relationships between values (a square table is a table whose length equals its width). As an example, the latter class can be expressed in OWL 1.1 using the axiom:

Class(SquareTable complete super(Table) restriction(holds(equal length width)).

Meta-modelling: Meta-modelling, i.e., the treatment of classes, properties and other entities as individuals, is allowed in some representation languages, including the OWL Full species of OWL, but was forbidden in OWL DL because of the computational difficulties that it may cause. However, users often say that they want some aspects of meta-modelling, at least the ability to associate simple information with classes such as synonyms, names in different languages, responsible person, etc. Annotation properties were added to OWL to partly satisfy this requirement, but the limited meta-modelling facililities provided by annotation properties have not satisfied users, particularly as annotation properties cannot be range-restricted.

OWL 1.1 allows a basic form of meta-modelling, called *punning*. In OWL 1.1, a name (like Person) can be used as any or all of an individual, a class, or a property. In fact OWL 1.1 relaxes the disjointness of vocabularies restriction in OWL DL in the cases in which it is possible to *disambiguate* the exact use of a name (for example, it is always possible to detect whether a name is used as a class or as an individual, but not always whether a property is used as an object or as a datatype property). For example, the following sequence of axioms is allowed in OWL 1.1: Class(Person), Individual(Person), Individual(John Person), SameIndividualAs(Person, Rock); however, these axioms do *not* entail Individual(John Rock).

Semantic-free comments: Annotations provide for the ability to include what might otherwise be considered "comments", such as information about the author or version number of a class. In OWL, however, this information carries semantics, and some counter-intuitive aspects, such as a membership in a class not being inferable simply because the class has a different version number. This has lead to the desire to have true comments, i.e., information associated with classes, etc., that has no semantic import at all. The following example shows some of the places that comments can appear:

```
Class(Person partial comment('The basic class for people'))
   super(Animal comment('Why not Mammal?'))
   type(MyClasscomment('My classes are special'))
   value(dc : creator peter comment(dc : creator john))
```

3 Influences on OWL 1.1

OWL 1.1 has borrowed heavily from recent research on Description Logics as well as from recent research on the nature of the Semantic Web.

3.1 OWL 1.1 and Description Logics

There have recently been two streams of work on extensions to the Description Logic underlying OWL DL (a.k.a. $\mathcal{SHOIQ}$).

Firstly, there has been considerable work on how best to add datatypes and relationships between data values to OWL-like languages. The general ideas and requirements are basically similar, but the various proposals differ in detail: the datatypes themselves need to be "admissible" (roughly speaking, this means that datatype predicates are closed under negation and that the satisfiability of conjunctions of these predicates is decidable), which ensures that we can use datatype solvers as blackboxes in OWL reasoners.

Secondly, extensions to expressive description logics allowing more expressive property constructs have been devised and investigated. This line of work has lead to the $\mathcal{RIQ}$ [11], and $\mathcal{SROIQ}$ [7] description logics, and effective reasoning processes for them.

The existence of this work in the Description Logic community has made it simple to add qualified number restrictions, enhanced property constructs, and more expressive datatypes.

3.2 OWL 1.1 and OWL

OWL 1.1 and OWL DL: As OWL 1.1 is a simple extension to OWL DL, it borrows heavily from OWL DL. To this end, OWL 1.1 uses the same basic syntax style as the so-called "abstract" syntax for OWL DL [16]. As well as using the same syntactic style, OWL 1.1 incorporates the entire OWL DL syntax, only providing extensions to it. In this way, any legal OWL DL ontology is also a legal OWL 1.1 ontology.

As well, the meaning of OWL 1.1 is compatible with the meaning of OWL DL. Instead of providing a direct model-theoretic semantics, the meaning of OWL 1.1 is provided by a mapping to the Description Logic $\mathcal{SROIQ}$ [2]. This method of providing a semantics for OWL 1.1 gives more direct access to the theoretical results concerning $\mathcal{SROIQ}$, and is foreshadowed by the work of Horrocks and Patel-Schneider reducing OWL DL entailment to Description Logic satisfiability [8].

OWL 1.1 and OWL Full: OWL 1.1 does not provide any significant features provided by OWL Full over OWL DL. This is largely because OWL 1.1 is essentially a Description Logic, and the facilities provided by OWL Full over OWL DL (meta-modelling, blending objects and datatypes, unusual syntactic forms, subverting basic constructs, etc.) are essentially those that go outside of the Description Logic paradigm.

The only significant aspect of OWL Full that shows up in OWL 1.1 is meta-modelling. However, OWL 1.1 provides meta-modelling facilities via *punning*, which is not compatible with the meta-modelling features of OWL Full (which are the same as those provided by RDF). See Section 3.3 for more on how meta-modelling distinguishes OWL 1.1 from RDF and OWL Full.

OWL 1.1 and OWL Lite: Expressive ontology languages, such as OWL 1.1 and OWL DL, though decidable, have a high worst-case computational complexity[3] and are hard to use and implement efficiently. The design of simpler ontology languages with more tractable inferences was considered of primary importance by the W3C Web Ontology Working Group. The OWL Lite subset of OWL DL was designed as a language that is easier to use and present to naive users, as well as easier to implement.

The Web Ontology Working Group concluded that the main complexity of OWL DL relies on the possibility of defining complex boolean descriptions using, for example, union and complement; as a consequence, OWL Lite explicitly prohibits unions and complements in the definition of concepts; additionally, OWL Lite limits all descriptions in the scope of a quantifier to concept names, does not allow individuals to show up as concepts, and limits cardinalities to 0 and 1. The goal was to significantly reduce the number of available modeling constructs, on the one hand, and to eliminate the major sources of non-determinism in reasoning, on the other hand.

Although OWL Lite looks much simpler than OWL DL, it is still possible to express more complex concept descriptions by introducing new concept names, exploiting implicit negations and using axioms to associate multiple descriptions with a given concept name. So, from a user perspective, OWL Lite is even harder to use than OWL DL, since the available modeling constructs do not correspond to the actual expressivity of the language. Also, from a computational perspective, OWL Lite is only slightly less complex than OWL DL (namely ExpTime-complete instead of NExpTime-complete), and all the important reasoning problems remain intractable.

[3] Satisfiability and subsumption are NExpTime-complete for $\mathcal{SHOIQ}$, and ExpTime-complete for $\mathcal{SHIQ}$.

OWL 1.1 identifies various subsets of OWL 1.1, each of which benefits from tractable (i.e., polynomial time) reasoning for one or more important reasoning tasks [3]. The intention is that these subsets can be used and efficiently implemented.

3.3 OWL 1.1 and RDF

OWL 1.1 diverges from the same-syntax extension of RDF vision of the Semantic Web, as embodied in RDFS and OWL Full. Like all species of OWL, OWL 1.1 uses URI references for its names and thus fits well into the Semantic Web. However, OWL 1.1 is not compatible with RDF, and thus is not compatible with OWL Full. There are two areas of incompatibility.

OWL 1.1 includes semantic-free comments. In RDF, as in OWL Full, all information is in the form of triples, and all triples have semantic import. This makes it impossible to include syntactic-only comments that can survive transmission or processing.

OWL 1.1 uses a (weak) form of meta-modelling called *punning*. In punning, names can be used for several purposes; for example, *Person* can at the same time be the name of a class and the name of an individual. The different uses of a name are, however, completely independent, and from a semantic point of view they can be thought of as separate names, e.g., *Person-the-Class* and *Person-the-Individual*.

Punning is compatible with annotation properties as used in OWL DL, as annotation properties were expressly designed so that their use would not have any effect on class level entailments. However, punning is not compatible with the meta-modelling possibilites inherent in the semantics of RDF [6] (and thus inherent in OWL Full), precisely because it makes the two uses of a name semantically independent.

A triple syntax is being provided for OWL 1.1, syntactically compatible with the triple syntax for OWL DL. However, for the above reasons, this syntax could not be given a meaning compatible with the RDF meaning for triples, at least not without some very difficult semantic tiptoeing (as well as some questionable encoding (such as creating fresh URI references for punning purposes, e.g., using *Person-the-Class* and *Person-the-Individual* instead of just *Person*). The appropriateness of continuing along this line with OWL 1.1 is called even more into question by the impossibility of extending it to Semantic Web languages with expressive power on a par with that of First-Order Logic [15].

4 OWL 1.1 Specification

OWL 1.1 is a complete logic, and thus come with a syntax and a (model-theoretic) semantics. Actually OWL 1.1 has two different syntaxes, the one described here and an XML syntax. We only consider here the extensions to the OWL DL abstract syntax [16].

4.1 Syntax for OWL 1.1

The "abstract" syntax of of OWL 1.1 is an extension of the "abstract" syntax for OWL DL. In this Section, we provide some examples on how the OWL DL syntax has been extended; for a complete specification we refer the reader to http://owl1-1.cs.manchester.ac.uk/syntax.html. OWL 1.1 provides additional expressive power to OWL DL, namely: qualified cardinality restrictions, local reflexivity restrictions for simple properties, reflexive, irreflexive, and antisymmetric flags for simple properties, disjointness of simple and datatype properties, and *regular* property inclusion axioms. As an explample, we show below the new syntax rules for cardinality restrictions and complex property inclusion axioms:

individualRestrictionComponent ::= *individualCardinality*
individualCardinality ::= minCardinality(*non-negative-integer description*)
| maxCardinality(*non-negative-integer description*)
| cardinality(*non-negative-integer description*)
axiom ::= SubPropertyOf(propertyChain(*individualvaluedPropertyID*$^+$)
individualvaluedPropertyID)

OWL 1.1 also provides syntactic sugar. For example, OWL 1.1 provides a construct that allows to state that an individual does not have a particular property value:

value ::= valueNot(*individualvaluedPropertyID individualID*)
| valueNot(*individualvaluedPropertyID individual*)
| valueNot(*datavaluedPropertyID dataLiteral*)

OWL 1.1 includes its own methods for user-defined datatypes, using a syntax similar to the one used in Protégé. The semantics for OWL 1.1 user-defined datatypes is taken from XML Schema Datatypes [1]. OWL 1.1 allows restrictions that relate values for different data-valued properties on the same individual.

restriction ::= holds(*datatypePredicateID* { *argument* })
restriction ::= *datatypePropertyID* | *dataLiteral*
datatypePredicateID ::= equal | notEqual | lessThan | lessThanEqual
| greaterThan | greaterThanEqual

OWL 1.1 allows arbitrary comments to be inserted in ontologies.

comment ::= Comment({ *dataLiteral* | *URIreference* })

A comment is allowed anywhere white space is allowed. Comments have no semantic import in OWL 1.1, but comments should survive processing and transmission by OWL 1.1 systems.

4.2 Semantics for OWL 1.1

The semantics for OWL 1.1 rely on a translation into the description logic $\mathcal{SROIQ}(\mathcal{D}^+)$, which extends the logic $\mathcal{SROIQ}$ [7] with datatypes and datatype restrictions. A similar translation was used to define the semantics of Standard OIL in terms of the Description Logic $\mathcal{SHIQ}(D)$ [4].

Since OWL 1.1 is an extension of OWL-DL (in the same way that $\mathcal{SROIQ}(\mathcal{D}^+)$ is an extension of $\mathcal{SHOIN}(\mathcal{D})$), this document also provides a well-defined semantics for OWL-DL documents that is equivalent to the direct model-theoretic semantics given in the OWL documentation [16]. Although both semantics are equivalent, a translation-based semantics has several advantages with respect to a direct semantics: a translation-based approach results in a cleaner, simpler and more precise specification; it gives direct access to theoretical results for the logic; and it provides a direct implementation pathway.

In the OWL 1.1 Semantics document [4] we define a *translation* function that maps OWL 1.1 ontologies into equivalent $\mathcal{SROIQ}(\mathcal{D}^+)$ knowledge bases. The translation function and the semantics of $\mathcal{SROIQ}(\mathcal{D}^+)$ [7] completely specify the semantics of OWL 1.1. The translation of OWL 1.1 into $\mathcal{SROIQ}(\mathcal{D}^+)$ is quite straightforward, and follows naturally from the syntax and semantics of OWL-DL and from the syntax and informal specification of OWL 1.1 given in Section 4.1. We refer the reader to the Semantics document for details on the translation.

5 Implementation

OWL 1.1 has been developed outside any formal standardization process. Instead, the intent was to advance the state of the deployed and used art before moving to a standards body. Several of the OWL 1.1 extensions were selected because they were already supported by some OWL tools and were deployed (or would quickly be deployed) by key users. For example, qualified number restrictions are supported by Racer, KAON2, and FaCT++ as well as the Protégé editor. Unfortunately, this support is primarily through the DIG interface and obsolete exchange formats.[5] Both these features are strongly in demand from the user community[21], but they are not used due to the lack of interoperability. Since this interoperability was mostly a matter of agreeing on a common syntax, it is likely that these features will be widely available after OWL 1.1 is finalized.

One difference in OWL 1.1 is the change in syntax. In OWL 1.1, there is a normative XML syntax that is described by an XML schema [6]and which is based on DIG 2.0. The WebOnt working group did produce a document describing a direct XML syntax for OWL, but it is incomplete and was never significantly used. We expect that the availability of a sensible XML schema friendly format

[4] http://owl1-1.cs.manchester.ac.uk/Semantics.html
[5] See http://www.w3.org/2001/sw/BestPractices/OEP/QCR/ for a discussion.
[6] http://owl1-1.cs.manchester.ac.uk/XMLsyntax.html

will make it possible to build useful OWL 1.1 tools based on the XML infrastructure. For example, schema aware editors could be fruitfully used to edit OWL 1.1, and XPath and XSLT could be used for a variety of tasks. There is also an RDF encoding of OWL 1.1 (thus, an RDF/XML exchange format for it), so users can adopt the format that best suits their needs. Some future extensions, however, may build on the XML format (see section 6).

Several categories of OWL 1.1 features (syntactic sugar, semantic-free comments, meta-modelling by punning) are essentially trivial to implement, since they can be handled with a transformation into the core formalism. From an implementation perspective, the most substantial extensions in OWL 1.1 are the property constructors. In particular, the known decision procedure for $\mathcal{SROIQ}$ involves the use of automata to manage the property chains. While the automata seem modular, there is only very limited experience with implementing and optimizing algorithms incorporating them [11]. The other extensions with regard to properties (e.g., disjoint roles or negated property assertions), while conceptually simpler (e.g., negated property roles may be encoded using nominals), also lack implementation experience. The consensus of implementors at the OWLED workshop was that these features were reasonable to implement, but the first implementations are not yet available.

Significantly, at the OWLED workshop, the major OWL reasoner implementors (those of Cerebra, RACER, FaCT++, KAON2, and Pellet) and editor implementors (Protégé and Swoop) pledged to support OWL 1.1 in a timely manner (in particular, for preliminary implementation within six months of reasonably firm specifications), and implementation work is already underway.

6 Future Extensions

OWL 1.1 was from the start intended to be an easy, incremental improvement to OWL. However, OWL 1.1 was also intended to start movement toward a larger extension of OWL, which, for the purposes of this paper, we shall refer to as OWL 2.0. There is a wide variety of academic and industrial research concerning expressive extensions to OWL, much of it driven by user demand, some of it driven by standardization in related areas. While it is difficult to predict what a future working group might find compelling, there are five obvious features which would be sensible to consider for the next version of OWL.

Syntactic extensibility: Since the OWLED workshop, there have been a number of additional proposals for syntactic sugar even beyond what OWL 1.1 offers. This suggests that some form of macro system would be useful. An obvious proposal is to center the system on the new XML syntax and make use of the extensive transformation infrastructure for XML (e.g., XPath and XSLT). Such a proposal is likely to emerge from the next OWL workshop.

Query: There are efficient implementations of some form of conjunctive ABox querying in Racer, Pellet, and KAON2. While the Data Access Working Group only defined the semantics of SPARQL queries for RDF graphs, there is a hook

allowing one to plug in other semantics, for example, that of OWL. It would be straightforward to support such in OWL 2.0.

Integration with rules: Integration of rules of various sorts and DL-based ontology languages is not only a hot research area, but also a requirement for the new Rules Interchange Format (RIF) Working Group. The OWL community could define some extensions to RIF specifically designed around OWL, e.g., based on SWRL [9] or on decidable variations of SWRL.

Non-monotonicity: A common request for OWL is non-montonic constructs. Unfortunately, in spite of the intense interest, there is little settled consensus or practical experience with non-monontonic features in DL systems. It may be that in the coming year the picture will become clearer, but there needs to be a more effective gathering of grounded use cases for non-monotonicity in OWL so that the appropriate design decisions can be made.

Meta-modelling: OWL 1.1 meta-modelling does not facilitate domain modelling [14], nor does it cover some useful sorts of annotative behavior. Although the meta-modelling facilities of OWL Full were strongly argued for within the WebOnt working group, actual use of *those* particular facilities is rare. So, more work must be done to determine what additional meta-modelling capabilities are both feasible and will be actually used.

References

1. Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001.
2. B. Cuenca-Grau. OWL 1.1 Web Ontology Language Model-theoretic Semantics. *Draft Document*, 2005.
3. B. Cuenca-Grau. Tractable Fragments of the OWL 1.1 Web Ontology Language. *Draft Document*, 2005.
4. Dieter Fensel, Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, and Peter F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.
5. Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of IJCAR 2001*, 2001.
6. Patrick Hayes. RDF model theory. W3C Recommendation, 10 February 2004.
7. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR-06*, 2006. To appear.
8. Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4):345–357, 2004.
9. Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of WWW 2004*, 2004.
10. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1):7–26, 2003.
11. Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proc. of IJCAI 2003*, 2003.

12. A. Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. SWOOP: a web ontology editing browser. *J. of Web Semantics*, 4(2), 2005.
13. T. Liebig and O. Noppens. Ontotrack: Combining browsing and editing with reasoning and explaining for owl ontologies. In *Proc. of ISWC 2004*, 2004.
14. Boris Motik. On the properties of metamodeling in OWL. In *Proc. of ISWC 2005*, 2005.
15. Peter Patel-Schneider. Building the semantic web tower from RDF straw. In *Proc. of IJCAI 2005*, 2005.
16. Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004.
17. Protégé. http://protege.stanford.edu/, 2003.
18. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. To appear, 2005.
19. K. Spackman. Managing clinical terminology hierarchies using algorithmic calculation of subsumption: Experience with SNOMED-RT. *J. of the Amer. Med. Informatics Ass.*, 2000. Fall Symposium Special Issue.
20. Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, 2004.
21. K. Wolstencroft, A. Brass, I. Horrocks, P. Lord, U. Sattler, D. Turi, and R. Stevens. A little semantic web goes a long way in biology. In *Proc. of ISWC 2005*, 2005.