D2.5.5 Next Steps for OWL

Bernardo Cuenca Grau
Ian Horrocks
Bijan Parsia
Peter Patel-Schneider
Ulrike Sattler
Rob Shearer

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.5.5 (WP2.5)
OWL 1.1 is a simple extension to 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.
Keyword list: description logics, ontology language, RDF, OWL DL, OWL, W3C, standardization

Document Identifier	KWEB/2006/D2.5.5/v1.0
Project	KWEB EU-IST-2004-507482
Version	v1.0
Date	17 July, 2006
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Fax: +43(0)5125079872, Phone: +43(0)5125076485/88
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polythechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Fax: +41 21 6935225, Phone: +41 21 6932738
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Fax: +33 2 99124098, Phone: +33 2 99124223
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Fax: +49 30 83875220, Phone: +49 30 83875223
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Fax: +39 0471 315649, Phone: +39 0471 315642
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Fax: +33 4 7661 5207, Phone: +33 4 7661 5366
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Fax: +30-2310-464164, Phone: +30-2310-464160
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Fax: +49-511-7629779, Phone: +49-511-76219711
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Fax: +353 91 526388, Phone: +353 87 6826940
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Fax: +44 1908 653169, Phone: +44 1908 653506
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Fax: +34-913524819, Phone: +34-913367439
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Fax: +49 721 6086580, Phone: +49 721 6083923
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Fax: +44(151)7943715, Phone: +44(151)7943667
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer Science,
University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Fax: +44 161 2756204, Phone: +44 161 2756248
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Fax: +44 114 2221810, Phone: +44 114 2221891
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Fax: +39 0461 882093, Phone: +39 0461 881533
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Fax: +31842214294, Phone: +31204447731
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Fax: +32 2 6293308, Phone: +32 2 6293308
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Changes

Version	Date	Author	Changes
1.0	14-07-2006	Rob Shearer	creation

Executive Summary

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. 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. Wide usage experience has revealed deficiences in the original design of the OWL standard, including certain limitations on expressiveness, lack of support for metamodelling, and onerous syntax for some common constructs. A number of use cases highlighting some of these problems (as well as proposed solutions) have been documented in KnowledgeWeb deliverable 2.5.4.

At the first "OWL: Experiences and Directions" workshop it was decided to design an extension to the OWL DL species of OWL which:

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.

Such an extension, called OWL 1.1, was designed based on user experiences and research in description logics which has been conducted since the original OWL standardization activity. The philosophy behind this updated version of the OWL DL language, as well as its syntax and semantics, is described in this deliverable. The relationships between OWL 1.1 and prior standards, including OWL and RDF, are explored.

Certain aspects of OWL 1.1 are considered trivial to integrate into existing OWL-DL systems and are expected to receive immediate wide support. All major tool vendors have committed to OWL 1.1 support in the near future.

Certain limitations of the OWL standard are not addressed by OWL 1.1, including query languages, integration with rules, non-monotonic extensions, and robust metamodelling features. Future directions and a longer-term vision of "OWL 2.0" are presented.

Contents

Chapter 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 [PSHH04]. 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 [HPSvH03]. The design of OWL was heavily influenced by research in description logics (DLs); investigations of (combinations of) DL language constructors provided a detailed understanding of the semantics and computational properties of, and reasoning techniques for, various ontology language designs [BCM$^+$03, HST99, HS01, HS05]; this understanding was used to ensure that, for two of the three OWL dialects (OWL DL and OWL Lite), key reasoning problems would be decidable. Basing the language on a suitable DL also allowed for the exploitation of existing DL implementations in order to provide reasoning services for OWL applications [Hor98, PS98, HM01].

The standardisation of OWL has led to the development and adaptation of a wide range of tools and services. These include reasoners such as FaCT++ [TH04], Racer [HM01] and Pellet [SPC$^+$05], and editing tools such as Protégé [Pro03], Swoop [KPS$^+$05], Ontotrack [LN04] and OilEd [BHGS01]. Editing tools typically use a reasoner to compute the class hierarchy, alert users to problems such as inconsistent classes, and answer queries; several now include sophisticated debugging tools such as explanation (of inconsistency and subsumption) [KPSH05].

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., OBO (http://obo.sourceforge.net/) and 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 in-

dustries. 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 [Mot05].

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.

Chapter 2

Overview

The initial design of 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.

For these reasons, it was decided at the first "OWL: Experiences and Directions" workshop (http://www.mindswap.org/2005/OWLWorkshop/) to design an extension to the OWL DL species of OWL called OWL 1.1—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 [PL94, Spa00, Rec02], e.g., it may be useful to express the fact that certain locative properties are transfered across certain part-whole properties so that, when using a medical terminology ontology, a trauma or 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.

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 carry-on bag is one where the sum of height, width, and depth does not exceed 45 inches).

Meta-modelling: Meta-modelling, specifically the ability to treat 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 facilities provided by annotation properties have not satisfied users, particularly as annotation properties cannot be range-restricted. A more general ability to annotate classes and properties using the full language vocabulary is desired, even if semantic restrictions must be imposed to avoid computational difficulties.

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 has semantic import, 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.

Chapter 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

OWL DL is based on a description logic called $\mathcal{SHOIN}$. Even when OWL DL was designed, there were discussions as to whether it should be based on $\mathcal{SHOIN}$ or on $\mathcal{SHOIQ}$ [HS05], the latter being the former's extension with *qualifying number restrictions*. This expressive means is rather useful for modeling [WBH+05], and is known to be no more problematic for a reasoner than the unqualified number restrictions in OWL DL. Interestingly, an effective decision procedure for $\mathcal{SHOIN}$ and $\mathcal{SHOIQ}$ has only been designed recently [HS05], but is already implemented in reasoners for OWL DL. Additionally, there have recently been two streams of work on extensions to the Description Logic underlying OWL DL.

Firstly, there has been considerable work on how best to add datatypes and relationships between data values to OWL-like languages [Lut03, PH05]. While the various proposals differ in detail, the general ideas and requirements are basically similar: 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 $\mathcal{RIQ}$ [HS03], $\mathcal{SRIQ}$ [HKS05] and $\mathcal{SROIQ}$ [HKS06] 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. OWL 1.1 essentially takes this work in its entirety and without significant modification.

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 "abstract" syntax for OWL DL [PSHH04]. 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 $\mathcal{SROIQ}$ [CG05a]. 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 [HPS04a].

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[1] 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

[1] Satisfiability and subsumption are NExpTime-complete for $\mathcal{SHOIQ}$, and ExpTime-complete for $\mathcal{SHIQ}$ [Tob01].

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 [Tob01]), and all the important reasoning problems remain intractable.

In contrast to OWL, OWL 1.1 does not single out just one language subset. Instead, various subsets of OWL 1.1 have been identified, each of which benefits from tractable (i.e., polynomial time) reasoning for one or more important reasoning tasks [CG05b]. The intention is that these subsets can be used and implemented as appropriate to a particular application.

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 by 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 [Hay04] (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 dif-

ficult semantic tiptoeing (such as providing comprehension principles for comments that essentially added every possible comment to every element of the domain of discourse) 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 [PS05].

Chapter 4

OWL 1.1 Specification

OWL 1.1 is a complete logic, and thus comes with a syntax and a (model-theoretic) semantics. Actually OWL 1.1 has two different syntaxes, the one described here and an XML syntax. The "abstract" syntax of OWL 1.1 is an extension of the "abstract" syntax for OWL DL. The core grammar for the language syntax is thus given by the original OWL DL specification [PSHH04]. In this document we provide additions to this grammar (new production rules most of which reference the pre-existing nonterminals) which extend the syntax to OWL 1.1.

4.1 Syntax for OWL 1.1

The extensions in OWL 1.1 lift its expressive power to that of SROIQ [HKS06]. This amounts to adding qualified cardinality restrictions, as an extension to restrictions on datatype properties and object properties; local reflexivity restrictions for simple properties, as an extension to restrictions on object properties; reflexive, irreflexive, and antisymmetric flags for simple properties, as an extension to the flags allowed on object properties; and disjointness of simple and datatype properties, and *regular* property inclusion axioms, as new axioms.

dataRestrictionComponent ::= *dataCardinality*
dataCardinality ::= minCardinality(*non-negative-integer dataRange*)
| maxCardinality(*non-negative-integer dataRange*)
| cardinality(*non-negative-integer dataRange*)
individualRestrictionComponent ::= *individualCardinality*
individualCardinality ::= minCardinality(*non-negative-integer description*)
| maxCardinality(*non-negative-integer description*)
| cardinality(*non-negative-integer description*)
individualRestrictionComponent ::= self
individualvaluedPropertyFlags ::= Reflexive | Irreflexive

| Symmetric | AntiSymmetric
axiom ::= DisjointProperties(*datavaluedPropertyID*$^+$)
| DisjointProperties(*individualvaluedPropertyID*$^+$)
axiom ::= SubPropertyOf(propertyChain(*individualvaluedPropertyID*$^+$)
individualvaluedPropertyID)

Only simple properties (i.e., properties that are not implied by property chains, see [HKS06] for details) can: have the self restriction component; be specified as being Reflexive, Irreflexive, Symmetric, or Antisymmetric; or be used in DisjointProperties axioms for individual-valued properties.

The SubPropertyOf axioms involving individual-valued properties must be *regular*. That is, there must be a strict partial order $<$ on individual-valued properties such that for each SubPropertyOf axiom involving individual-valued properties, of the form SubPropertyOf(*S R*) *S* is the inverse of *R*, *S* is of the form propertyChain(*R R*), *S* is of the form propertyChain(*S1 ... Sn*) and each *Si* $<$ R, *S* is of the form propertyChain(*R S1 ... Sn*) and each *Si* $<$ *R*, or *S* is of the form propertyChain(*S1 ... Sn R*) and each *Si* $<$ *R*.

The first couple of other additions in OWL 1.1 are simple syntactic sugar. To make the common construct of multiple disjoint classes easier to state, OWL 1.1 provides an axiom that directly states that a group of classes are pairwise disjoint, instead of having to use separate disjoint axioms for each pair of classes. Similarly, OWL 1.1 provides a construct allowing the direct assertion that an individual does not have a particular property value, instead of, e.g., having to state that the individual is an instance of a suitable restriction class.

axiom ::= DisjointUnion(*description*$^+$)
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 [BM01].

dataRange ::= datatype(*datatypeID* { *datatypeRestriction* })
datatypeRestriction ::= *datatypeFacet(dataLiteral*)
datatypeFacet ::= length | minLength | maxLength | pattern | enumeration
| maxInclusive | maxExclusive | minInclusive | minExclusive
| totalDigits | fractionDigits
axiom ::= Datatype(*datatypeID* { *annotation* } base(*datatypeID*)
{ *datatypeRestriction* })

Datatype facets should only by used where they would be allowed in XML Schema Datatypes, except that the length, minLength, maxLength, and pattern facets are not

allowed for numeric types. Datatype facets have the same meaning as in XML Schema Datatypes, except that they uniformly work in the value space, never the lexical space. If a datatype facet is used in a way that has no meaning, such as (length 5∧∧xsd:string), then the datatype extension is empty.

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

The syntax here allows an arbitrary number of arguments, but must be appropriate for the predicate, and all the current predicates only allow two arguments. All invalid combinations are unsatisfiable (i.e., they do not signal an error). The equality and order for a particular base type is taken from XML Schema Datatypes. If a base datatype does not have an order then the ordering restriction is unsatisfied.

OWL 1.1 removes the limitation imposed in OWL DL that the sets of class, individual and property names must be pairwise disjoint. The semantic change to allow this without computational consequences is to break the RDF-inspired connection between class and property extensions and the individual denotation of names. With this change, any name can be made the subject of a non-annotation property, but in this (syntactic) context the name is always (semantically) interpreted as an individual. As simple syntactic sugar, non-annotation properties can be used where annotations are allowed in OWL DL.

annotation ::= *value* | type(*description*)

A class or property axiom with an annotation is syntactic sugar for an extra Individual axiom relating the class or property name to the annotations.

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

While the OWL-DL specification provides direct model-theoretic semantics for the language, the semantics for OWL 1.1 rely on a translation into the description logic $\mathcal{SROIQ}(\mathcal{D}^+)$,

OWL 1.1 Abstract Syntax	$\mathcal{SROIQ}(\mathcal{D}^+)$ Syntax
A (Class URI)	Concept name A
owl:Thing	$\top$
owl:Nothing	$\bot$
R (Object Property URI)	Abstract role R
U (Datatype Prop. URI)	Concrete role U
a (Individual URI)	Individual a
ϕ (Data Value or Plain Literal)	Concrete value ϕ
p_m (Datatype Predicate ID)	Concrete Predicate p_m
Φ (OWL 1.1 Datatype)	Supported Datatype Φ
intersectionOf(C_1...C_n)	$\sigma(C_1) \sqcap ... \sqcap \sigma(C_n)$
unionOf(C_1...C_n)	$\sigma(C_1) \sqcup ... \sqcup \sigma(C_n)$
complementOf(C)	$\neg\sigma(C)$
oneOf(a_1 .... a_n)	$\{\sigma(a_1)\} \sqcup ... \sqcup \{\sigma(a_n)\}$
oneOf(ϕ_1...ϕ_n)	$\{\sigma(\phi_1), ..., \sigma(\phi_n)\}$
restriction(R X_1...X_n)	σ(restriction(R X_1)) $\sqcap$ ... $\sqcap$ σ(restriction(R X_n))
restriction(R someValuesFrom(C))	$\exists\sigma(R).\sigma(C)$
restriction(R allValuesFrom(C))	$\forall\sigma(R).\sigma(C)$
restriction(R hasValue(a))	$\exists\sigma(R).\{\sigma(a)\}$
restriction(R self)	$\exists\sigma(R).$Self
restriction(R minCardinality(n))	$\geq n\sigma(R).\top$
restriction(R maxCardinality(n))	$\leq n\sigma(R).\top$
restriction(R Cardinality(n))	$= n\sigma(R).\top$
restriction(R minCardinality(n C))	$\geq n\sigma(R).\sigma(C)$
restriction(R maxCardinality(n C))	$\leq n\sigma(R).\sigma(C)$
restriction(R Cardinality(n C))	$= n\sigma(R).\sigma(C)$
restriction(U X_1...X_n)	σ(restriction(U X_1)) $\sqcap$ ... $\sqcap$ σ(restriction(U X_n))
restriction(U someValuesFrom(Φ))	$\exists\sigma(U).\sigma(\Phi)$
restriction(U allValuesFrom(Φ))	$\forall\sigma(U).\sigma(\Phi)$
restriction(U hasValue(ϕ))	$\exists\sigma(U).\{\sigma(\phi)\}$
restriction(U minCardinality(n))	$\geq n\sigma(U)$
restriction(U maxCardinality(n))	$\leq n\sigma(U)$
restriction(U Cardinality(n))	$= n\sigma(U)$
restriction(U minCardinality(n Φ))	$\geq n\sigma(U).\sigma(\Phi)$
restriction(U maxCardinality(n Φ))	$\leq n\sigma(U).\sigma(\Phi)$
restriction(U Cardinality(n Φ))	$= n\sigma(U).\sigma(\Phi)$
restriction(holds(p_m) U_1...U_m))	$\exists\sigma(U_1)...\sigma(U_m).\sigma(p_m)$

OWL 1.1 Abstract Syntax	$\mathcal{SROIQ}(\mathcal{D}^+)$ Syntax
ObjectProperty(R super(R_1)...super(R_m))	$\bigcup_{i=1}^{m} \{\sigma(R) \sqsubseteq \sigma(R_i)\}$
domain(C_1)...domain(C_m)	$\bigcup_{i=1}^{m} \{\geq 1\sigma(R) \sqsubseteq \sigma(C_i)\}$
range(C_1)...range(C_m)	$\bigcup_{i=1}^{m} \{\top \sqsubseteq \forall\sigma(R).\sigma(C_i)\}$
inverseOf(S)	$\{\sigma(R) \equiv \sigma(S)^-\}$
[Symmetric]	$\{\sigma(R) \equiv \sigma(R)^-\}$
[Functional]	$\{\top \sqsubseteq \leq 1\sigma(R)\}$
[InverseFunctional]	$\{\top \sqsubseteq \leq 1\sigma(R)^-\}$
[Transitive]	{Trans(σ(R))}
[Reflexive]	{Ref(σ(R))}
[Irreflexive]	{Irr(σ(R))}
[AntiSymmetric]	{ASymm(σ(R))}
SubPropertyOf(R_1 R_2)	$\{\sigma(R_1) \sqsubseteq \sigma(R_2)\}$
SubPropertyOf(propertyChain($R_1 \cdots R_m$) R)	$\{\sigma(R_1)...\sigma(R_m) \sqsubseteq \sigma(R)\}$
DisjointProperties(R_1 R_2)	{Dis(σ(R_1),σ(R_2))}
EquivalentProperties(R_1...R_m)	$\bigcup_{i=1}^{m-1} \{\sigma(R_i) \equiv \sigma(R_{i+1})\}$
DataProperty(U super(U_1)...super(U_m)	$\bigcup_{i=1}^{m} \{\sigma(U) \sqsubseteq \sigma(U_i)\}$
domain(C_1)...domain(C_m)	$\bigcup_{i=1}^{m} \{\geq 1\sigma(U) \sqsubseteq \sigma(C_i)\}$
range(Φ_1)...range(Φ_m)	$\bigcup_{i=1}^{m} \{\top \sqsubseteq \forall\sigma(U).\sigma(\Phi_i)\}$
[Functional])	$\{\top \sqsubseteq \leq 1\sigma(U)\}$
SubPropertyOf(U_1 U_2)	$\{\sigma(U_1) \sqsubseteq \sigma(U_2)\}$
EquivalentProperties(U_1...U_m)	$\bigcup_{i=1}^{m-1} \{\sigma(U_i) \equiv \sigma(U_{i+1})\}$
DisjointProperties(U_1 U_2)	{Dis(σ(U_1),σ(U_2))}
Class(A partial C_1...C_m)	$\{\sigma(A) \sqsubseteq \sigma(C_1) \sqcap ... \sqcap \sigma(C_m)\}$
Class(A complete C_1...C_m)	$\{\sigma(A) \equiv \sigma(C_1) \sqcup ... \sqcup \sigma(C_m)\}$
EnumeratedClass(A a_1...a_m)	$\{\sigma(A) \equiv \{\sigma(a_1)\} \sqcup ... \sqcup \{\sigma(a_m)\}\}$
EquivalentClasses(C_1...C_m)	$\bigcup_{i=1}^{m-1} \{\sigma(C_i) \equiv \sigma(C_{i+1})\}$
DisjointClasses(C_1...C_m)	$\bigcup_{i,j=1;i\neq j}^{m} \{\sigma(C_i) \sqsubseteq \neg\sigma(C_j)\}$
DisjointUnion(C C_1...C_m)	$\{\sigma(C) \equiv \sigma(C_1) \sqcup ... \sqcup \sigma(C_m)\} \cup \bigcup_{i,j=1;i\neq j}^{m} \{\sigma(C_i) \sqsubseteq \neg\sigma(C_j)\}$
Individual(a type(C_1)...type(C_m)	$\bigcup_{i=1}^{m} \{\sigma(C_i)(\sigma(a))\}$
value(R_1 b_1)...value(R_m b_m)	$\bigcup_{i=1}^{m} \{\sigma(R_i)(\sigma(a), \sigma(b_i))\}$
value(U_1 ϕ_1)...value(U_m ϕ_m)	$\bigcup_{i=1}^{n} \{\sigma(U_i)(\sigma(a), \sigma(\phi_i))\}$
valueNot(R_1 b_1)...valueNot(R_m b_m)	$\bigcup_{i=1}^{n} \{\neg\sigma(R_i)(\sigma(a), \sigma(b_i))\}$
valueNot(U_1 ϕ_1)...valueNot(U_m ϕ_m))	$\bigcup_{i=1}^{m} \{\neg\sigma(U_i)(\sigma(a), \sigma(\phi_i))\}$
SameIndividual(a_1....a_m)	$\bigcup_{i,j=1;i\neq j}^{m} \{\sigma(a_i) \doteq \sigma(a_j)\}$
DifferentIndividuals(a_1...a_m)	$\bigcup_{i,j=1;i\neq j}^{m} \{\sigma(a_i) \neq \sigma(a_j)\}$

Table 4.1: Translation of OWL 1.1 into $\mathcal{SROIQ}(\mathcal{D}^+)$

which extends the logic $\mathcal{SROIQ}$ [HKS06] with datatypes and datatype restrictions. A similar translation was used to define the semantics of Standard OIL in terms of the Description Logic $\mathcal{SHIQ}(D)$ [FvHH+01].

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 [PSHH04]. 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.

We will 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 $\mathcal{SROIQ}(\mathcal{D}^+)$ completely specify the semantics of OWL 1.1. The semantics of $\mathcal{SROIQ}$, along with a decision procedure for reasoning in the logic, are given in [HKS06]; the semantics of a suitable datatype extension are given in [PH05].

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. Let $\mathcal{O}_0$ be an OWL 1.1 ontology and let $\{\mathcal{O}_j\}_{1\leq j\leq m}$ be the set of ontologies imported (directly or indirectly) by $\mathcal{O}_0$; let $\{\alpha_1^j, ..., \alpha_{n_j}^j\}$ for $0 \leq j \leq m$ be the set of axioms and facts contained in $\mathcal{O}_j$. The translation into a $\mathcal{SROIQ}$ knowledge base $\mathcal{T}$ is as follows: $\mathcal{T} = \bigcup_{0\leq j\leq m} \sigma(\mathcal{O}_j)$, where $\sigma(\mathcal{O}_j) = \bigcup_{1\leq i\leq n_j} \sigma(\alpha_i^j)$. The syntactic correspondence between OWL 1.1 descriptions and $\mathcal{SROIQ}(\mathcal{D}^+)$ concepts is given in Table 4.1 as is the correspondence between OWL 1.1 axioms and facts and $\mathcal{SROIQ}(\mathcal{D}^+)$ axioms. This table completely specifies the translation function σ and should be read as follows: given a construct in OWL 1.1 abstract syntax in the first column, its evaluation under σ is given in the second column.

Chapter 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.[1] Similarly, Pellet will reason with user defined datatypes, but it will not consume the format Protégé emits. Both these features are strongly in demand from the user community[WBH+05], 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 [2]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 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 chapter 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 particu-

[1] See http://www.w3.org/2001/sw/BestPractices/OEP/QCR/ for a discussion.

[2] http://homepages.cs.manchester.ac.uk/seanb/dig/overview.html

lar, 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 [HS03]. 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.

Chapter 6

Future Extensions

OWL 1.1 was from the start intended to be an easy, incremental improvement to OWL. With a year and a half of experience with OWL, there was strong consensus on several of the obvious holes in the language. 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 [PS06], 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 [HPS04b] or on decidable variations of SWRL [MSS04, Ros05].

Non-monotonicity: A common request for OWL is non-montonic constructs. Unfortu-

nately, in spite of the intense interest, there is little settled consensus or practical experience with non-monontonic features in description logic 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 [Mot05], nor does it cover some useful sorts of annotative behavior [GW04]. 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 [Wan06]. So, more work must be done to determine what additional meta-modelling capabilities are both feasible and will be actually used. Clearly the first three classes of feature are more ripe for standardisation than the last two. Of course, there are many other features one might hope for in the next version of OWL, for example, even more expressive datatypes, role constructors, fixed point operators and hybrid logic constructs.

Chapter 7

Conclusion

We have presented OWL 1.1, an incremental extension to OWL DL that exploits recent developments in DL languages and reasoning techniques in order to satisfy some common requirements expressed by users of OWL DL. Although some of these extensions are no more than syntactic sugar, others add real expressive power to the language, and in particular significantly extend what can be said about properties. In spite of this increased expressive power, OWL 1.1 retains the desirable computational properties of OWL DL: key reasoning problems are decidable, and practical decision procedures are available for them. Support from the implementors of prominent OWL DL reasoning and editing tools means that OWL 1.1 compatible systems should be available in the near future.

Although OWL 1.1 is backwards compatible with OWL DL, it departs significantly from OWL DL in some important respects: the semantics of OWL 1.1 is not given directly, but via a mapping to $\mathcal{SROIQ}(\mathcal{D}^+)$; OWL 1.1 uses XML Schema for its normative exchange syntax; the RDF syntax of OWL 1.1 does not fully respect the semantics of RDF, and so is not semantically compatible with OWL Full.

It is anticipated that OWL 1.1 will be only a first step, and that larger extensions will follow. These could include support for, e.g., (some form of) rules, macro and query languages. It is also anticipated that, given sufficient support from implementors and users, it may be appropriate to initiate standardisation activities for OWL 1.1 and/or OWL 2.0.

Bibliography

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BHGS01] Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: A Reason-able ontology editor for the semantic web. In *Proc. of KI*, 2001.

[BM01] Paul V. Biron and Ashok Malhotra. XML schema part 2: Datatypes. W3C Recommendation, May 2001.

[CG05a] B. Cuenca-Grau. OWL 1.1 Web Ontology Language Model-theoretic Semantics. *Draft Document*, 2005.

[CG05b] B. Cuenca-Grau. Tractable Fragments of the OWL 1.1 Web Ontology Language. *Draft Document*, 2005.

[FvHH+01] 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), 2001.

[GW04] N. Guarino and C. Welty. An overview of ontoclean. In *Handbook of Ontologies*. 2004.

[Hay04] Patrick Hayes. RDF model theory. W3C Recommendation, 10 February 2004.

[HKS05] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The irresistible $\mathcal{SRIQ}$. In *Proc. of the First OWL Experiences and Directions Workshop*, 2005.

[HKS06] Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *KR-06*, 2006. To appear.

[HM01] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of IJCAR*, 2001.

[Hor98] Ian Horrocks. The FaCT system. In *Proc. of TABLEAUX*, 1998.

[HPS04a] Ian Horrocks and Peter Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. of Web Semantics*, 1(4), 2004.

[HPS04b] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of WWW*, 2004.

[HPSvH03] 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), 2003.

[HS01] Ian Horrocks and Ulrike Sattler. Ontology reasoning in the $\mathcal{SHOQ}$(D) description logic. In *Proc. of IJCAI*, 2001.

[HS03] Ian Horrocks and Ulrike Sattler. Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. In *Proc. of IJCAI*, 2003.

[HS05] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of IJCAI*, 2005.

[HST99] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR*, 1999.

[KPS+05] 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.

[KPSH05] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in owl ontologies. *J. of Web Semantics*, 3(4), 2005.

[LN04] T. Liebig and O. Noppens. Ontotrack: Combining browsing and editing with reasoning and explaining for owl ontologies. In *Proc. of ISWC*, 2004.

[Lut03] C. Lutz. Description logics with concrete domains—a survey. In *Advances in Modal Logics Vol. 4*. World Scientific Publ., 2003.

[Mot05] Boris Motik. On the properties of metamodeling in OWL. In *Proc. of ISWC*, 2005.

[MSS04] B. Motik, U. Sattler, and R. Studer. Query answering in OWL-DL with rules. In *Proc. of ISWC*, 2004.

[PH05] Jeff Pan and Ian Horrocks. OWL-Eu: Adding customised datatypes into OWL. In *Proc. of ESWC*. 2005.

[PL94] Lin Padgham and Patrick Lambrix. A framework for part-of hierarchies in terminological logics. In *Proc. of KR*, 1994.

[Pro03] Protégé. http://protege.stanford.edu/, 2003.

[PS98] P. F. Patel-Schneider. DLP system description. In *Proc. of the 1998 Description Logic Workshop*, 1998.

[PS05] Peter Patel-Schneider. Building the semantic web tower from RDF straw. In *Proc. of IJCAI*, 2005.

[PS06] Eric Prud'hommeaux and Andy Seaborne. SPARQL query language for RDF. W3C Candidate Recommendation, 6 April 2006.

[PSHH04] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. OWL web ontology language semantics and abstract syntax. W3C Recommendation, 10 February 2004.

[Rec02] A. Rector. Analysis of propagation along transitive roles: Formalisation of the galen experience with medical ontologies. In *Proc. of the 2002 Description Logic Workshop*, 2002.

[Ros05] Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *J. of Web Semantics*, 3(1), 2005.

[Spa00] 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.

[SPC+05] E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. To appear, 2005.

[TH04] Dmitry Tsarkov and Ian Horrocks. Efficient reasoning with range and domain constraints. In *Proc. of the 2004 Description Logic Workshop*, 2004.

[Tob01] S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.

[Wan06] T.D. Wang. Gauging ontologies and schemas by numbers. In *WWW06 Workshop on Evaluation of Ontologies for the Web*, 2006. To Appear.

[WBH+05] 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.