Pushing the limits of OWL, Rules and Protégé
A simple example

Anne Cregan[1,2], Malgorzata Mochol[3], Denny Vrandečić[4], and Sean Bechhofer[5]

[1] University of New South Wales, Australia, annec@cse.unsw.edu.au
[2] National ICT Australia (NICTA)
[3] Freie Universität Berlin, Germany, mochol@inf.fu-berlin.de
[4] AIFB, Universität Karlsruhe (TH), Germany, denny@aifb.uni-karlsruhe.de
[5] University of Manchester, UK, sean.bechhofer@manchester.ac.uk

Abstract. The Semantic Web brings powerful languages for creating models, based on Description Logics and Rules. These languages are used within ontology engineering tools and applied in strong and efficient reasoners. Working within the context of a Summer School, we explored these technologies by creating a small and easy to understand example, working firstly with OWL alone, and introducing SWRL rules as needed, to automatically classify a number of instances according to some intuitively simple criteria. We present the example OWL ontology, SWRL rules, and the issues and problems we encountered. Our experience highlights the capabilities and limitations of OWL and of SWRL, not in theoretical but in practical terms, and points to the need for better tool support, but is primarily a lesson in "traps for young players" in terms of formulating classifications, that we hope will be instructive for Semantic Web students following in our footsteps, and their tutors.

1 Introduction

The Semantic Web[4] vision is to extend the Web with machine-understandable data, forming a global distributed knowledge store which application may leverage to perform tasks automatically. The base technologies for realizing this vision are: Uniform Resource Identifiers(URIs) as a global identification mechanism for resources; the Resource Description Framework(RDF) as a basic data model, together with its XML-based serialisation syntax for publishing data on the Web[16]; the Web Ontology Language(OWL) which extends RDF with more expressive knowledge representation[15, 14]; and a yet to be defined rule language, for which the Semantic Web Rule Language(SWRL) may be considered a prototype[7]. The development of the Semantic Web is a joint effort of scientific and business institutions around the globe, led by the World Wide Web Consortium(W3C).

This paper describes the experiences of a group of students(Cregan, Mochol, Vrandečić) and their tutor(Bechhofer) in a mini-project conducted at the Third European Summer School on Ontological Engineering and the Semantic Web(SSSW05) held in Spain in July 2005[1]. The students, all PhD students at

[1] `http://babage.dia.fi.upm.es/sssw05/`

their respective institutions, formed a project group with the aim of discovering the limits of OWL in relation to the use of rules. Specifically, we wanted to discover what tasks are beyond the expressivity of OWL and require the use of rules, and what additional capabilities do rules provide? We used Protégé ontology software[12], the Racer automated reasoner[3] and SWRL in our investigation: SWRL is based on a combination of the OWL DL and OWL Lite sublanguages of OWL together with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language.

To facilitate the investigation, we devised a simple example problem which we regard as a realistic, though small, use case for Semantic Web technologies. The paper describes the challenges that arose and our learning process in applying various conditions to our example domain. We hope that our experience will be helpful both as a teaching resource, adding to the (currently) limited amount of tutorial material available for teaching Semantic Web technologies to students (see [6, 11, 13]), and also, by highlighting the problems we experienced, will help tool implementers and standards bodies to identify issues inherent in their approaches, and initiate discussion of possible solutions. The material discussed is available on `http://www.aifb.uni-karlsruhe.de/WBS/dvr/rove`. It may also be useful as an early test case for reasoners or editors implementing SWRL.

The paper is structured as follows: Firstly, we specify the general scenario of our use case in section 2. Then we describe the ontology we constructed, and present some limitations of the expressivity of OWL in section 3, and of SWRL in section 4. Section 5 deals with problems we experienced using the tools, and outlines some directions for future work. Lastly in section 6, we summarise our learning experience, the issues we encountered and their significance.

2 Scenario

All participating students at SSSW05 were asked to form a group of four or five students to conduct a mini-project. At the outset, the organizers asked the students to make the groups as mixed as possible, to ensure everyone had a diverse collaborative experience contrasting with their usual work/study activities. The organizers also impressed on the students that having fun was a very important part of the activity (no doubt as it enhances learning!). Every student belonged to exactly one group. Each group had a unique name, and was led by exactly one summer school tutor. Every tutor led at least one group, and some led more than one group. All participants in the mini-project were either tutors or students, and no-one was both a student and a tutor.

Due our chosen area of investigation we named our mini-project group "*ROVE*" (Rules for Ontology Validation Effort). For the ROVE project, we chose a simple and readily accessible domain for applying ontology representation and rules: the student groups taking part in the Summer School mini-projects. In order to discover the limits of OWL's abilities, and the capabilities provided by adding rules, we attempted to formally define and implement the informally stated conditions the organizers had placed on group formation.

2.1 Conditions

We decided that the following conditions reflected desirable group formations:
Condition #1 : Groups should have either 4 or 5 members
Condition #2 : Groups should have at least one member of each gender
Condition #3 : Group members are of all different nationalities
Condition #4 : Group members are from all different institutions
Condition #5 : Groups should have fun. We decided it would be fun if the tutor of the group were the favourite of all the students in the group, so we asked all students to nominate which tutor was most attractive (see sec 4).
Our goal was to formalize these conditions and use a classifier to automatically categorize all groups as either a *GoodGroup* - one that fulfills all conditions, or a *BadGroup* - one which does not satisfy one or more of the stated conditions:

$$\text{Group is a } \mathit{GoodGroup}\ iff\ \mathit{Cond}1 \wedge \mathit{Cond}2 \wedge \mathit{Cond}3 \wedge \mathit{Cond}4 \wedge \mathit{Cond}5$$
$$\text{Group is a } \mathit{BadGroup}\ iff \neg\mathit{Cond}1 \vee \neg\mathit{Cond}2 \vee \neg\mathit{Cond}3 \vee \neg\mathit{Cond}4 \vee \neg\mathit{Cond}5$$

3 ROVE Ontology

3.1 Classes and Instances

Initially we built a simple ontology (see Fig. 1) containing disjoint classes *Person*, *Country*, *Institution*, and *Group*. *Person* was divided into disjoint subclasses *Tutor* and *Student* exactly partitioning the class. *Person* was also divided by gender into disjoint subclasses *Man* and *Woman*, also as an exact partition.

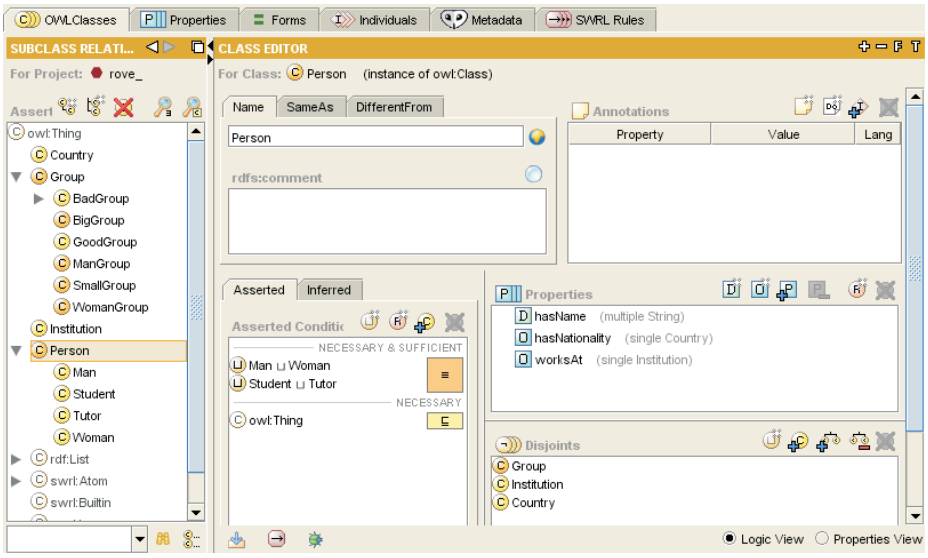

Fig. 1. ROVE-ontology

3.2 Properties

Object Properties were then set up as follows: each person has a Nationality (*hasNationality*), and works at an Institution(*worksAt*). Each Group is led by a Tutor (*ledBy*) and has members (*hasMember*), only from the class *Student*. Each Student is a *memberOf* exactly one Group, and has a favorite Tutor (*attractedTo*). Data Properties related Persons and Groups to name strings.

3.3 Asserting Conditions using OWL

Using this ontology, we then tried to formalise and implement as many of our stated conditions as possible using only OWL expressivity.

Condition #1 : Groups should have either 4 or 5 members

This condition was easily implemented by setting minimum and maximum cardinalities on the property *hasMember* which related Groups to Students:

$$Group \sqsubseteq \geq 4\, hasMember \sqcap \leq 5\, hasMember$$

As all the groups satisfied this condition and cardinality conditions are easily formulated within OWL, we simply asserted this condition on Group.

Condition #2 : Groups should have at least one member of each gender

This condition could not be modelled in OWL in the same way as Condition #1, as it requires *hasMember* to have a minimum cardinality for values from each of the subclasses *Man* and *Woman*. We used existential restrictions to accomplish the task, by defining *GoodGroup* as a subclass of *Group* where *hasMember* has *(owl:)someValuesFrom Man* and *(owl:)someValuesFrom Woman*. However, we wanted to capture that this condition was only one of those to be satisfied by a good group, so we did not want to make it a sufficient condition. Yet we also wanted to ensure that all members of the group must be either male or female, and there was no way to do this using a necessary condition.
We then approached the problem in reverse by specifying conditions for being a bad group rather than a good group. Note that not satisfying any one of our five conditions is sufficient for classification as a bad group. Protégé supports explicit specification of either 'Necessary', or 'Necessary and Sufficient' asserted conditions (see Figure) through an interface box, but if one has conditions which are Sufficient without being Necessary, as in this case, this can only be implemented by adding a subClass. We felt it would be more intuitive for Protégé to support the assertion of subsumption axioms through the interface box also.
By introducing two new concepts to represent groups with all male members or all female members: *ManGroup* and *WomanGroup*, where:

$$WomanGroup \equiv Group \sqcap \forall hasMember.Woman$$
$$ManGroup \equiv Group \sqcap \forall hasMember.Man$$

and creating BadGroup as a subclass of Group, it was then simple to implement these two sufficient conditions for being a "bad group" by making *WomanGroup* and *ManGroup* subclasses of *BadGroup*:

$$WomanGroup \sqsubseteq BadGroup$$
$$ManGroup \sqsubseteq BadGroup$$

However, having set this up, we noticed in automatic classification that one of the groups was not classified as a *BadGroup*, although it consisted of four male students. This was due to OWL's Open World Assumption: the group, having four male members, could still potentially have a fifth member who may be female, without breaking the cardinality restriction, thus the reasoner could not classify the group as a *BadGroup*. To solve this problem we considered several alternatives. We had to dismiss the possibility of defining groups by enumeration, as this included nominals and was not supported by the available reasoners [5]. Had we pursued this path, we would have needed to rework our ontology to model group membership as a class instead of a relation, and this did not seem intuitive to us in any case. Instead, we decided to state the size of the groups explicitly by creating two new concepts: *BigGroup* (group with exactly 5 members) and a *SmallGroup* (group with exactly 4 members) as disjoint subclasses of Group, and then assert our other conditions onto these concepts:

$$BigGroup \equiv Group \sqcap \geq 5\, hasMember \sqcap \leq 5\, hasMember$$
$$SmallGroup \equiv Group \sqcap \geq 4\, hasMember \sqcap \leq 4\, hasMember$$
$$Group \equiv SmallGroup \sqcup BigGroup$$

Conditions #3-#5

These conditions required consideration of more than one property at a time: for example, both a student's nationality and group membership. Whilst OWL has a relatively rich set of class constructors, expressivity for properties is much weaker. Whilst OWL permits chaining of properties, it does not support making assertions about the equality of the objects at the end of two different properties/property chains. Since conditions #3-#5 require precisely this kind of assertion, it was not possible to formulate them using only OWL.

4 ROVE Rules

In the next step, we explored the use of rules to implement our conditions, using the Semantic Web Rule Language(SWRL)[7] plugin to Protégé with the ROVE ontology described above. Rules are constructed in the form of an implication between an antecedent (body) and a consequent (head): whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. In SWRL rules, one may assert equivalences as well as implications. SWRL provides Horn-like rules for both OWL DL and OWL Lite, includes a high-level syntax for representing these rules[1] and is more powerful than either OWL DL or Horn rules alone[8].

Condition #3 : Groups should have members of all different nationalities

To classify bad groups in regard to this condition we constructed a rule "SameNationalitiesRule" which states that if a group has any two members with the same nationality it is a *BadGroup*. Specifically, if group g has member s and member s has nationality n and group g has another member x who is different from member s and member x also has nationality n then g is a *BadGroup*. SWRL notation:

$$hasMember(?g, ?s) \wedge hasNationality(?s, ?n) \wedge hasMember(?g, ?x) \wedge hasNationality(?x, ?n) \wedge differentFrom(?s, ?x) \rightarrow BadGroup(?g)$$

Condition #4 : Groups should have members from all different institutions

The same approach applies as was used for different nationalities:

$$hasMember(?g, ?s) \wedge worksAt(?s, ?i) \wedge hasMember(?g, ?x) \wedge worksAt(?x, ?i) \wedge differentFrom(?s, ?x) \rightarrow BadGroup(?g)$$

Condition #5 : Groups should have fun

To stress-test OWL's abilities to represent and reason with compositions of properties and situations where there were multiple properties that connected the classes, we devised a complex criteria for"fun group": a fun group is one where all the students in the group are attracted to the tutor leading their group. We also had the ulterior motive of providing amusement for both tutors and students when the project work was presented on the last day of the summer school.
Each student was asked which tutor they were most attracted to: if the student asked for clarification as to what "attracted to" meant, they were told that they may interpret it as they wished. As the school had more male tutors than female (5 as opposed to 2) and more male students than female (35 to 20), then assuming a traditional interpretation of "attraction" there is likely to be a bias in the data, favoring the student body's attraction to the female tutors, at least in terms of raw numbers. However, while the results were a source of much amusement for the participants (the ROVE presentation included poll results for "Most attractive Summer School tutor"), they were inconsequential for the real purpose of the exercise (although perhaps not to the tutors!).
We formulated the rule to capture that any group that has a member who is attracted to someone other than the group's tutor is a bad group:

$$hasMember(?g, ?s) \wedge attractedTo(?s, ?t) \wedge hasTutor(?g, ?x) \wedge differentFrom(?t, ?x) \rightarrow BadGroup(?g)$$

Incidentally, our other ulterior motive in formulating all the conditions was to surreptitiously ensure that the ROVE group was the only group that satisfied all conditions, and we are pleased to say that we successfully achieved this.

4.1 OWL: Not Bad does not equal Good

At this point we thought we had now captured all our conditions adequately using OWL and SWRL, and would be able to automatically classify the groups as good or bad. Whilst we were able to classify *BadGroup*s, we had no way do the converse: that is, to automatically classify *GoodGroup*s. Stating that *GoodGroup*s are all groups that are not bad groups will not work: not being classified as a bad group only indicates that the group's status as a good or bad group is unknown. There is no way to specify that the list of criteria is exhaustive and use negation as failure, as OWL's semantics adopt the open world assumption: a statement cannot be assumed true just because its negation cannot be proven[9].
Therefore, in order to classify *GoodGroup*s as such, we needed to reformulate all the prior conditions in a positive form, thereby setting up necessary and sufficient conditions for being a *GoodGroup*. By positive form, we mean a formulation designed to satisfy the condition, rather than to violate it. For instance, with the nationality condition (Condition #3), the positive form ensures all members of the Group have different nationalities, whereas the negative form simply tests whether any two members have the same nationality.
We found it particularly challenging to grasp the asymmetric implications of using the positive and negative forms of rules, as intuitively we were only attempting to view the situation from the opposite side. We felt that a symmetric approach would be easier to use, as could be provided by tools, for example a wizard creating closures over rules automatically. Maybe such a pattern should be expressible in SWRL itself, in order to increase interoperability.
By formulating positive rules for our conditions, we were able to define *GoodGroup*s as exactly those Groups which satisfied all the rules:

$$GoodGroup \equiv MixedGenderGroup \sqcap InternationalGroup \sqcap InterInstitutionalGroup \sqcap FunGroup$$

This is done by creating an OWL axiom using class descriptors defined in SWRL. However, formulating the rules in positive form in SWRL was extremely onerous for most of our conditions.

Condition #2 : Groups should have at least one member of each gender

Creating a *MixedGenderGroup* was easy: we were able to take the pure-OWL approach previously described in section 3.3:

$$MixedGroup \equiv Group \sqcap \exists hasMember.Male \sqcap \exists hasMember.Female$$

Condition #3 : Groups should have members of all different nationalities

To define this condition positively, we constructed an *InternationalGroup* rule containing as antecedent a pairwise comparison of nationalities of every member to test whether they were all different. There need to be two forms of the rule for *BigGroup* and *SmallGroup*, as 5-member groups require 5 different nationalities, not 4 : since every binary combination has to be considered, 4-member

groups require only 6 pairwise comparisons, whilst 5-member groups require 10 comparisons ($\sum_1^{n-1} n$ comparisons for n members).
To show the complexity of such a rule we have shown it below for a 3-member group only:

Part of the rule	Explanation	Meaning
$SmallGroup(?x)$	if x is a small group	
$\wedge$	and	
$hasMember(?x, ?a) \wedge$ $hasMember(?x, ?b) \wedge$ $hasMember(?x, ?c)$	group x has members: a, b and c	Group x has 3 members.
$\wedge$	and	
$differentFrom(?a, ?b) \wedge$ $differentFrom(?a, ?c)$	member a is different from members b and c	Group x has 3 **different** members
$\wedge$	and	
$differentFrom(?b, ?c)$	member b is different from member c (equality to a was checked above)	
$\wedge$	and	
$hasNationality(?a, ?e) \wedge$ $hasNationality(?b, ?f) \wedge$ $hasNationality(?c, ?g)$	member a has nationality e and member b has nationality f and member c has nationality g	Each group member has nationality (no information regarding different or same nationalities).
$\wedge$	and	
$differentFrom(?e, ?f) \wedge$ $differentFrom(?e, ?g)$	nationality e is different from nationalities f and g	Each of the 3 group members has **different** nationalities.
$\wedge$	and	
$differentFrom(?f, ?g)$	nationality f is different from nationality g (equality to e was checked above)	
$\rightarrow$	then	
$GoodGroup(?x)$	the group x is a good group	

Such a rule is long and unwieldy to write with the SWRL plugin: we lamented the lack of pre-defined predicates in SWRL, as a simple "all-different" predicate to enter the arguments and have SWRL take care of the pairwise comparisons would have been a great aid.

Condition #4 : Groups should have members from all different institutions

Analogously to the above, *InterInstitutionalGroup* is another huge rule, requiring separate forms for 4 and 5 member groups.

Condition #5 : Groups should have fun

Although the intuition behind the definition of a *FunGroup* was easy - a group where all its members were attracted to the every tutor leading the group - formalizing the rule in positive form again proved to be an extremely tedious task, resulting in a huge and hard to maintain rule.

5 Tool support

The first and most important problem was the lack of an appropriate reasoner that could be plugged into a rule-enriched ontology created with Protégé. Racer supported reasoning within OWL, but to use rules in automatic classification obviously requires a reasoning engine which is able to operate on both OWL ontologies and SWRL rules, and none were available for the Windows-based

Summer School environment. However, we plan to use such reasoners, like *Hoolet*[2] or *KAON2*[10] and describe the results on the ROVE website.
Further problems related to the ontology engineering environment where we identified two issues with Protégé and one with SWRL:
Issue #1: entering data proved very tedious when making instances of a superclass belong to an orthogonal subclass partition: in our example, *Person* was partitioned into *Student/Tutor* subclasses as well as *Man/Woman*, and when entering the second SuperClass in Protégé we could not simply select a number of instances and assign them to a class, but had to edit each instance individually.
Issue #2: the tabs containing "necessary" and "necessary & sufficient" conditions left us wondering why there is no simple "sufficient" tab in Protégé as well to create subsumption axioms, as described in section 3.
Issue #3: as described in section 4.1, entering SWRL rules to create a closure for a class description with regard to a superclass was very tedious and error-prone. Even in our simple use case, the rules quickly become very large and therefore difficult to edit, let alone maintain and extend. A real-life example is likely to produce rules that are not manageable manually, so creating (and especially maintaining) such closures automatically needs to be made possible.

6 Conclusions

In this simple example of our Summer School mini-project, we showed the difficulties and challenges of implementing some intuitively very simple class constraints. The key learning experience were the limitations of the expressivity of OWL. By going beyond OWL and using SWRL, we were able to formulate several further conditions using rules, but found it challenging to fully understand the implications of the Open World Assumption and the lack of Negation as Failure. Without the input of our tutor, we might easily have fallen into the trap of believing we had captured all our conditions, when in actual fact we had not. Constructing and editing long rules in SWRL to address this was particularly difficult, and we strongly encourage developers to introduce better support for rule construction using templates and built-in predicates. We hope our experience will provide a lesson for Semantic Web students following in our footsteps, and will provide insights for Semantic Web tool developers. The ROVE material is available in different stages at `http://www.aifb.uni-karlsruhe.de/WBS/dvr/rove` for use in training and tool testing.

Acknowledgements We thank Antoine Zimmermann, who also was a member of the ROVE team, Enrico Motta and Asuncion Gomez-Perez, the organizers of the SSSW2005, and all the students, tutors and speakers. Congratulations to Natasha Fridman-Noy who was the Summer School's 'most attractive tutor'. Yes Asun, now it's time to go to the bus.
Research reported in this paper has been partially financed by NICTA (`www.nicta.com.au`), the EU project SEKT (`www.sekt-project.com`), the NoE KnowledgeWeb (`knowledgeweb.semanticweb.org`), and the German Ministry of Research (BMBF)

project Knowledge Nets (`nbi.inf.fu-berlin.de/research/wissensnetze`) of the InterVal (`interval.hu-berlin.de`) Berlin Research Centre for the Internet Economy (`www.internetoekonomie.net`).

References

1. R. Aggarwal. Semantic web services languages and technologies: Comparison and discussion. LSDIS Lab, University of Georgia, 2004. citeseer.ist.psu.edu/700682.html.
2. S. Bechhofer. Hoolet. Department of Computer Science, University of Manchester, 2004. http://owl.man.ac.uk/hoolet/.
3. V. Haarslev and R. Möller. Racer: An owl reasoning agent for the semantic web. In *Proc. of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with 2003 IEEE/WIC International Conference on Web Intelligence, Halifax Canada, Oct 13*, pages 91–95, 2003.
4. J. Hendler, T. Berners-Lee, and E. Miller. Integrating applications on the semantic web. *Journal of the Institute of Electronic Engineers of Japan*, pages 676–680, 2002.
5. J. Hladik. Reasoning about nominals with FaCT and RACER. In *Proc. of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, 2003.
6. M. Horridge, H. Knublach, A. Rector, R. Stevens, and C. Wroe. A practical guide to building OWL ontologies using the Protégé-OWL plugin and CO-ODE tools, 2004. University of Manchester.
7. I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M.Dean. Swrl: A semantic web rule language combining owl and ruleml. DARPA DAML Program, 2003. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.
8. I. Horrocks, P. F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. Owl rules: A proposal and prototype implementation. *Conditionally accepted for publication. Journal of Web Semantics*, 2005.
9. I. Horrocks, P. F. Patel-Schneider, and F. 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.
10. U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In D. Dubois, C. Welty, and M.-A. Williams, editors, *Proceedings of the KR2004*, pages 152–162. AAAI Press, 2004.
11. F. Manola and E. Miller. Resource Description Framework (RDF) Primer. W3C Recommendation 10 February, 2004. At http://www.w3.org/TR/rdf-primer/.
12. N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protégé-2000: Combining interoperability and flexibility. In R. Dieng and O. Corby, editors, *Proc. of the 12th EKAW*, volume 1937 of *LNAI*, pages 17–32. Springer, 2000.
13. A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe. OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In E. Motta, N. R. Shadbolt, and A. Stutt, editors, *Proc. of the 14th EKAW*, volume 3257 of *LNCS*, pages 63–81. Springer, 2004.
14. M. K. Smith, C. Welty, and D. McGuinness. OWL Web Ontology Language Guide, 2004. W3C Recommendation, http://www.w3.org/TR/owl-guide/.
15. W3C. Owl web ontology language-reference. LSDIS Lab, University of Georgia, 2004. http://www.w3.org/TR/owl-ref/.
16. W3C. W3c: Rdf/xml syntax specification (revised). LSDIS Lab, University of Georgia, 2004. http://www.w3.org/TR/rdf-syntax-grammar/.