Exploring OWL and rules: a simple teaching case

Malgorzata Mochol*

Freie Universität Berlin, Germany
E-mail: mochol@inf.fu-berlin.de
*Correspondong author

Anne Cregan

National ICT Australia (NICTA),
University of New South Wales, Australia
E-mail: annec@cse.unsw.edu.au

Denny Vrandečić

AIFB, Universität Karlsruhe (TH), Germany
E-mail: denny@aifb.uni-karlsruhe.de

Sean Bechhofer

University of Manchester, UK
E-mail: sean.bechhofer@manchester.ac.uk

Abstract: This study includes developing a suitable ontology to represent a simple domain using an ontology editor, representing logical constraints using constructs available within OWL-DL, and then illustrating that some of the classification criteria set for the task cannot be represented within OWL and require logical rules to be added using a rule language to achieve the desired functionality. An automated reasoner is used for classification and the impact of the Open World Assumption on classification results is examined. The case supports hands-on exercises using Semantic Web tools and languages to represent a simple domain, giving exposition of the underlying logical formalisms.

Keywords: semantic web; ontology; OWL; web ontology language; rules; reasoner; SWRL; semantic web rule language.

Reference to this paper should be made as follows: Mochol, M., Cregan, A., Vrandečić, D. and Bechhofer, S. (2008) 'Exploring OWL and rules: a simple teaching case', *Int. J. Teaching and Case Studies*, Vol.

Biographical notes: Malgorzata Mochol is a Graduate Research Assistant and Lecturer at the Institute for Computer Science of the Free University Berlin, Group of Networked Information Systems where she works primarily on ontology matching and application of Semantic Web technologies in e-business. She earned her degree in Computer Science at the Technical University Berlin in 2003.

Anne Cregan is a PhD student with the Knowledge Representation and Reasoning Group of the Artificial Intelligence Lab in the School of Computer Science and Engineering at the University of New South Wales, Australia. She is also sponsored by the National Centre of Excellence for Information and Communication Technologies, Australia (NICTA).

Denny Vrandečić is a researcher at the AIFB, working primarily on collaborative ontology engineering and evaluation. He received his Master in Computer Science and Philosophy at the University of Stuttgart.

Sean Bechhofer is a Lecturer in the School of Computer Science of the University of Manchester. He graduated in Mathematics from the University of Bristol in 1988 and has worked in Manchester since 1992, primarily in the area of tools and infrastructure to support the Semantic Web.

1 Summary of educational aspects of this paper

This case study provides practical teaching material for introducing undergraduate students to the essential aspects of Semantic Web languages, tools and reasoning. It provides subject matter and resources to provide student experience in designing, building and reasoning with ontologies, with emphasis on giving a concrete context for students to explore the interaction between ontologies as defined by the W3C's Web Ontology Language (OWL)[1], rules as defined by rule languages such as the Semantic Web Rule Language (SWRL)[2], and the impact of their underlying logical formalisms (e.g., OWL's Open World Assumption), in relation to achieving desired outcomes in automated classification using Description Logic Reasoning.

Currently available Semantic Web editors, tools and reasoners which may be applied to the task, and the authors' experiences with them, are described and compared. The study is intended to be used in conjunction with the materials provided online at http://km. aifb.uni-karlsruhe.de/projects/rove.

2 Introduction

This case study provides practical teaching materials for introducing undergraduate students to Semantic Web Languages, Tools and Reasoning. It provides:

- a description of a scenario and task which requires the use of Semantic Web technologies to achieve an automated classification according to set criteria
- a summary of currently available Semantic Web tools which may be used
- a possible solution to the task
- problems likely to be encountered
- links to other resources
- teaching guidelines.

All tools, materials and resources referred to are available online through the ROVE website at http://km.aifb.uni-karlsruhe.de/projects/rove.

2.1 Motivation

The key questions motivating this case study are:

- What are the limits of OWL's expressivity?
- When does OWL need to be supplemented with rules to provide additional functionality?
- What additional capabilities do rules provide?

It is not intended to answer these questions in a formal way – such answers are readily available – but rather to allow the students to test what can and what cannot be expressed with a certain language with regards to a specific use case, and thus increase their understanding about the limitations of the languages.

2.2 Learning outcomes

The material provided by this study gives practical hands-on experience in:

- designing an ontology to represent a simple domain
- using ontology editors (e.g., Protege, SWOOP) to build a simple ontology and populate it
- adding axioms to the ontology to capture logical properties of the domain
- using OWL-DL and understanding how its constructs represent ontologies and their logical constraints
- visualising and querying ontologies using tools
- using automated reasoners (e.g., Racer, Fact, Pellet) to perform automated classification tasks
- investigating the limits of the OWL ontology language: finding out what it can and can not express
- supplementing ontologies with rules in a rule language such as SWRL to provide additional expressivity
- investigating the behaviour of Description Logics and automated reasoners
- understanding the logical implications of OWL's 'Open World Assumption'.

The case study and questions posed provide an expose of current Semantic Web technologies and tools, and executing the task will provide students with a tangible illustration of some of the more elusive aspects of Description Logics.

2.3 Using this teaching case

Practical exercises based on this case study would work well as a precursor to, or in parallel with a more technical formal treatment of the Semantic Web field.

The case study itself is self-contained and does not require previous knowledge of Semantic Web technologies, although it is advanced enough to provide interest and challenges for those students who may have some previous experience in building and using ontologies. Some previous experience in formulating *Necessary and Sufficient* conditions would be advisable, although practical exercises based on this case study may be adapted to suit the level of the audience, from a step-by-step tutorial with tools provided, to a research-oriented task which simply sets out the required classification criteria and encourages students to find and explore various tools which are available to achieve the desired results.

2.4 Background

The material contained in this case study originated as a mini-project within the 3rd European Summer School on Ontological Engineering and the Semantic Web (SSSW05), held in Spain in July 2005. The mini-project, named "Rules for Ontology Validation Effort" (ROVE) was initiated by a group of post-graduate students to explore the limits of functionality of the Web Ontology Language (OWL), and the additional functionality which could be achieved by adding rules to OWL ontologies. The authors of this case study are three of the four post-graduate students of the 'ROVE' group (Cregan, Mochol and Vrandečić) and the group's tutor (Bechhofer), all PhD research students at their respective institutions, whilst Bechhofer, the group's tutor, has played a key role in authoring Semantic Web languages and tools for some time. The task was originally conducted using tools readily available at the time: Protégé with OWL-DL and SWRL Plug-in, used in conjunction with RACER for automated reasoning. A thorough description of the insights gained is given in Cregan et al. (2005).

2.5 Potential variations and local adaptations

Instance population and conditions

The ontology presented captures information about people and groupings of people, and the task set is to automatically classify each group of people according to whether it meets certain intuitively simple criteria imposed on group membership and structure. The knowledge domain is therefore readily understood, and the task easily grasped, so that the students can fully concentrate on exploring the representation itself.

The case was originally constructed with instance data provided by the students and tutors at the summer school undertaken by the authors, and pre-existing populated ontologies (partially anonymised for public availability) in various stages corresponding to amendments described in the paper, are available at the ROVE website http://km.aifb.uni-karlsruhe.de/projects/rove. However, for best results the students should be allowed to build their own ontologies from scratch, to give an appreciation of the design decisions involved.

The case study can easily be modified for local use by populating with details of the student group being trained, which is probably more interesting for the students and connects them better to the given tasks. In this case, the criteria set on group membership will usually need to be adapted to suit the students (e.g., using course enrolment instead of nationality, etc.). Creativity is of course encouraged: there are any number of possible conditions for testing which would be both instructive and potentially entertaining.

Tools.

The case study refers to currently available tools and languages (OWL, SWRL, Protege, etc), but as this is a developing field with tools and languages rapidly evolving, the exploration of new tools and standards becoming available is encouraged. The described task and variants on it could readily be conducted with any available ontology editor using some ontology language, rules, and an automated reasoner. In the original task, the authors experienced numerous technical problems using the available tools (further described in Section 4) and we believe this case study provides a useful base case for tool testing.[3]

2.6 Organisation of the case study

The case study is organised as follows: Section 3 describes the task set for the exercise including background of the scenario, the conditions for testing, and some key considerations in approaching the task. Section 4 contains a brief description of some available tools (editors and reasoners) which may be used to conduct the task. This is followed by the specification of a possible solution (Section 5) which shows 'step by step' how to satisfy the requirements of the task. The steps include building an ontology to cover as many of the stated conditions as possible within OWL, showing that it is not possible to capture all the requirements within the ontology itself, and then adding rules to capture the remaining conditions. The classification results highlight the impact of OWL's Open World Assumption, and a clear exposition of all the rules needed to complete the task gives a good insight into reasoning with Description Logics. Section 6 describes some practical problems likely to be encountered in using the tools and completing the task, and provides some suggestions for dealing with these. Throughout the case study, reference is made to various resources including reasoners, editors and languages, and Section 7 provides a list sort by category, including websites for each resource. The teaching guidelines are presented in Section 8 which is followed by the brief conclusion closing the case study in Section 9.

3 Description of the task

3.1 Scenario

The 3rd European Summer School on Ontological Engineering and the Semantic Web (SSSW05) was held in Spain for a week in July 2005. The student group at SSSW05 was made up of some 60 post-graduate students, both male and female, having many different nationalities and originating from many different educational institutions. At the outset, all participating students at SSSW05 were asked to form themselves into a group of four or five students to conduct a mini-project of their own choice. The summer school organisers asked the students to form groups as diverse as possible, in terms of having mixtures of nationalities, institutions and genders in each group. This was to provide a collaborative experience contrasting with usual work and study activities. The organisers also impressed on the students that having fun was a very important part of the activity (no doubt as it enhances learning!).

3.2 Goals and requirements

The authors, joined by Antoine Zimmerman, formed a project group with the intention of investigating the use of rules in conjunction with OWL. Sean Bechhofer, due to his expertise in the area, was invited to be the group's tutor. Concerning the chosen area of research, the mini-project group was named '*ROVE*'. 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 organisers had placed on group formation.

We decided that the following conditions reflected desirable group formations:

Condition #1: Every group should have either 4 or 5 members

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

Condition #3: Members of a group should all be different nationalities

Condition #4: Members of a group should all be from 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 to them.

Our goal was to formalise these conditions and use a classifier to automatically categorise all groups as either a `GoodGroup` – one that fulfills all the stated conditions, or a `BadGroup` – one which does not satisfy one or more of the stated conditions. Stated logically:

Group is a `GoodGroup` *iff* $Cond1 \wedge Cond2 \wedge Cond3 \wedge Cond4 \wedge Cond5$

Group is a `BadGroup` *iff* $\neg Cond1 \vee \neg Cond2 \vee \neg Cond3 \vee \neg Cond4 \vee \neg Cond5$

3.3 Ontology design and construction

In building an ontology there are potentially many design decisions to be made to ensure the ontology will best suit the stated purpose. There is not necessarily a 'right' and 'wrong' way to do it, but usually some designs will provide the desired functionality more readily than others.

The fundamental ontology constructs provided by OWL are classes, instances and properties, where:

- *classes* contain *instances*, e.g., the class `Person` contains specific individual `Mary`
- *classes* may have *subclasses* which inherit their characteristics; e.g., `Person` can be subclassed as `Male` or `Female`, where any individual male or female belongs to the class of `Person` as well as to the appropriate gender subclass
- *classes* may themselves be *subclasses* of *superclasses*, e.g., the class `Person` could belong to a superclass of `Animals`
- *instances* may have *properties* which connect them to specific values (*DataProperties*) or individuals, e.g., a person may have a specific age (data) and have a specific relationship to other individuals e.g., a person is the child of another person
- *property* relationships are at the individual level but logical conditions on them are defined at the class level; e.g., the property 'child of' from the previous example is defined as having a *domain*, *range* and *cardinality* which applies to any specific examples.

Note that in the examples given above, person's gender could be represented **either** as subclasses over the class `Person`, **or** as properties mapping each person instance to a value of `Male` or `Female`. Designing an ontology usually commences with choosing an appropriate domain representation which determines which entities in the domain will be treated as classes, properties and instances.

In our case, we began by constructing a simple ontology to contain data about the students, tutors and project groups. We determined firstly that we should build a class for 'Person' whose instances would correspond to individual people. As our desired classifications depended on characteristics of individual people within groups, we needed to capture each person's: gender, nationality, educational institution, membership of a mini-project group, status as a student or tutor, and if a student, which tutor was their favourite.

In order to determine the best representation, we needed to consider the logical properties of the domain which we would need to encode:

- all participants in the mini-project were individual persons
- all participants were either tutors or students
- no-one was both a student and a tutor
- all participants were either male or female
- no-one was both male and female
- every student belonged to exactly one group.
- each group had a unique name
- each group was led by exactly one summer school tutor
- every tutor led at least one group, and some led more than one group.

Some possible representations which adequately capture these conditions are suggested in Section 5, following a consideration of some of the Semantic Web tools available for approaching the task.

4 Available tools

One major advantage of Semantic Web technologies are their standardisation on interchange formats and thus their interoperability. In principle it should not matter which tools are used for editing the ontologies and later for reasoning with them, as long as they are using an OWL representation. This advantage (and its practical limitations) can be demonstrated in class by allowing the students to take tools of their own choice. In this section we will discuss some available and popular tools that can be used. We are aware that several further tools exist but cannot provide an extensive list here: furthermore as this is a rapidly evolving field, additional tools are regularly becoming available.

Download links and references to additional resources regarding these tools are given in Section 7. All the tools briefly described here are in active development at the time of writing, and it is recommended to use a recent version of each.

4.1 Ontology editors

Protégé

- *Protégé* is currently the best known ontology development environment and was used in the original development of the ROVE ontology. It has been in active development by the Stanford Medical Informatics (SMI) group of the University of Stanford since the early 1990s. A number of Semantic Web plug-ins have been developed specifically for use with Protégé. In particular, an OWL plug-in available for use with Protégé produces ontologies in the OWL language. This plug-in was developed largely by the Co-ode project in Manchester http://www.co-ode.org. The SWRL plug-in for Protégé supports the use of rules.
- The OWL and SWRL plug-ins provide enough expressivity to state all required axioms. The user interface of Protégé, although intuitive, is not based directly on OWL, so teaching OWL with Protégé requires consideration of how Protégé translates into the OWL language and conversely, how constructs within OWL are represented in Protégé. For example, the notion of "*Necessary and Sufficient conditions*" in the Protégé interface is translated to semantically equivalent OWL subsumption axioms, which may not appear obvious on the first glance. Protégé itself does not include a reasoner but may be used with any external reasoner which uses the Description Logic Implementation Group (DIG) (Bechhofer et al., 1999) interface. Protégé has an extensive range of plug-ins for visualisation and querying of ontologies, and exploration of these is encouraged.

SWOOP

- *SWOOP* is an ontology editor based on a browser-inspired interface, developed by the MINDSWAP group of the University of Maryland. By default, SWOOP ships with the Pellet reasoner, and is integrated more tightly with it than is possible via the DIG interface. It provides useful features like explanations (cf. Figure 1), which are especially helpful for learners. At the time of writing, the last stable release of SWOOP (Version 2.2) is now more than a year old, and does not offer support for rules. However, SWOOP is in active development and far more advanced releases are available, but users are warned that they may be unstable.

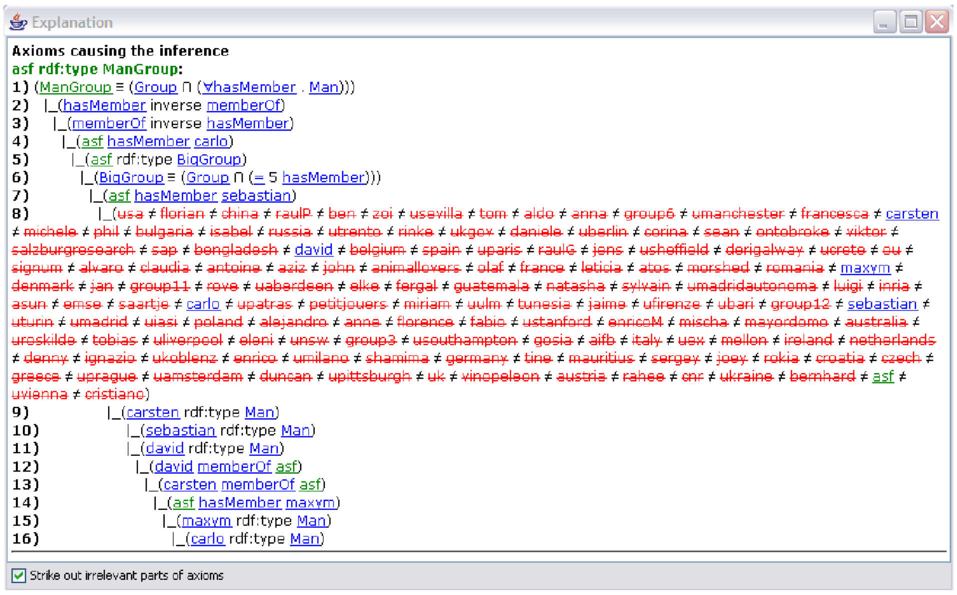
Both editors are freely available for download.

4.2 Reasoners

It is currently standard practice for ontology editors to use the DIG interface to enable connection to a reasoner of the user's choice. The following reasoners all offer a DIG interface, and can thus be combined with the editors as wished.

However, let us first note a regrettable limitation of the current DIG interface (Bechhofer et al., 1999), in that it does not specify how to exchange and reason over rules. Thus even though both the editor and the reasoner being used provide support for rules, they are not able to be used together seamlessly via the editor's interface. This problem is addressed in a proposal to extend the DIG interface appropriately (Bechhofer et al., 2006), but for now other solutions must be applied, like using a file based ontology exchange.

Figure 1 Swoop editor (see online version for colours)



Explanation

Axioms causing the inference
asf rdf:type ManGroup:
1) (ManGroup ≡ (Group ⊓ (∀hasMember . Man)))
2) |_(hasMember inverse memberOf)
3) |_(memberOf inverse hasMember)
4) |_(asf hasMember carlo)
5) |_(asf rdf:type BigGroup)
6) |_(BigGroup ≡ (Group ⊓ (= 5 hasMember)))
7) |_(asf hasMember sebastian)
8) |_(~~usa~~ ≠ ~~florian~~ ≠ ~~china~~ ≠ ~~raulP~~ ≠ ~~ben~~ ≠ ~~zoi~~ ≠ ~~usevilla~~ ≠ ~~tom~~ ≠ ~~aldo~~ ≠ ~~anna~~ ≠ ~~group6~~ ≠ ~~umanchester~~ ≠ ~~francesca~~ ≠ carsten ≠ ~~michele~~ ≠ ~~phil~~ ≠ ~~bulgaria~~ ≠ ~~isabel~~ ≠ ~~russia~~ ≠ ~~utrento~~ ≠ ~~rinke~~ ≠ ~~ukgov~~ ≠ ~~daniele~~ ≠ ~~uberlin~~ ≠ ~~corina~~ ≠ ~~sean~~ ≠ ~~ontobroke~~ ≠ ~~viktor~~ ≠ ~~salzburgresearch~~ ≠ ~~sap~~ ≠ ~~bengladesh~~ ≠ david ≠ ~~belgium~~ ≠ ~~spain~~ ≠ ~~uparis~~ ≠ ~~raulG~~ ≠ ~~jens~~ ≠ ~~usheffield~~ ≠ ~~derigalway~~ ≠ ~~ucreto~~ ≠ ~~eu~~ ≠ ~~signum~~ ≠ ~~alvaro~~ ≠ ~~claudia~~ ≠ ~~antoine~~ ≠ ~~aziz~~ ≠ ~~john~~ ≠ ~~animallovers~~ ≠ ~~olaf~~ ≠ ~~france~~ ≠ ~~leticia~~ ≠ ~~atos~~ ≠ ~~morshed~~ ≠ ~~romania~~ ≠ maxym ≠ ~~denmark~~ ≠ ~~jan~~ ≠ ~~group11~~ ≠ ~~rove~~ ≠ ~~uaberdeen~~ ≠ ~~elke~~ ≠ ~~fergal~~ ≠ ~~guatemala~~ ≠ ~~natasha~~ ≠ ~~sylvain~~ ≠ ~~umadridautonoma~~ ≠ ~~luigi~~ ≠ ~~inria~~ ≠ ~~asun~~ ≠ ~~emse~~ ≠ ~~saartje~~ ≠ carlo ≠ ~~upatras~~ ≠ ~~petitjouers~~ ≠ ~~miriam~~ ≠ ~~uulm~~ ≠ ~~tunesia~~ ≠ ~~jaime~~ ≠ ~~ufirenze~~ ≠ ~~ubari~~ ≠ ~~group12~~ ≠ sebastian ≠ ~~uturin~~ ≠ ~~umadrid~~ ≠ ~~uiasi~~ ≠ ~~poland~~ ≠ ~~alejandro~~ ≠ ~~anne~~ ≠ ~~florence~~ ≠ ~~fabio~~ ≠ ~~ustanford~~ ≠ ~~enricoM~~ ≠ ~~mischa~~ ≠ ~~mayordomo~~ ≠ ~~australia~~ ≠ ~~uroskilde~~ ≠ ~~tobias~~ ≠ ~~uliverpool~~ ≠ ~~eleni~~ ≠ ~~unsw~~ ≠ ~~group3~~ ≠ ~~usouthampton~~ ≠ ~~gosia~~ ≠ ~~aifb~~ ≠ ~~italy~~ ≠ ~~uex~~ ≠ ~~mellon~~ ≠ ~~ireland~~ ≠ ~~netherlands~~ ≠ ~~denny~~ ≠ ~~ignazio~~ ≠ ~~ukoblenz~~ ≠ ~~enrico~~ ≠ ~~umilano~~ ≠ ~~shamima~~ ≠ ~~germany~~ ≠ ~~tine~~ ≠ ~~mauritius~~ ≠ ~~sergey~~ ≠ ~~joey~~ ≠ ~~rokia~~ ≠ ~~croatia~~ ≠ ~~czech~~ ≠ ~~greece~~ ≠ ~~uprague~~ ≠ ~~uamsterdam~~ ≠ ~~duncan~~ ≠ ~~upittsburgh~~ ≠ ~~uk~~ ≠ ~~vinopeleon~~ ≠ ~~austria~~ ≠ ~~rahee~~ ≠ ~~cnr~~ ≠ ~~ukraine~~ ≠ ~~bernhard~~ ≠ asf ≠ ~~uvienna~~ ≠ ~~cristiano~~)
9) |_(carsten rdf:type Man)
10) |_(sebastian rdf:type Man)
11) |_(david rdf:type Man)
12) |_(david memberOf asf)
13) |_(carsten memberOf asf)
14) |_(asf hasMember maxym)
15) |_(maxym rdf:type Man)
16) |_(carlo rdf:type Man)

[✓] Strike out irrelevant parts of axioms

- *RACER* also known as RacerPro, is a commercial reasoner developed by RACER Systems. It is currently available for free for educational and scientific purposes, and requires registration. Registration may take a while, so make sure to have the licenses available before class.

  Although RacerPro provides almost complete support for OWL-DL it still has some limitations:

  - individuals in class expressions (so-called nominals) are only approximated (although at this time RacerPro is the only optimised OWL system that supports reasoning about individuals)

  - it can not currently process user-defined datatype types given as external XML Schema specifications (although all required datatypes of OWL-DL are properly supported)
  - RacerPro 1.9 does not employ the *Unique Name Assumption*, required by OWL-DL, i.e., it is not possible to state that two different URIs denote the same individual.

    However, UNA can be enabled globally to maximise efficiency. None of these limitations are issues for the exercises of this case study.

    Version 1.9 of RacerPro offers support and an integrated development environment for reasoning over ontologies enhanced with SWRL rules.

- *KAON2* is a reasoner developed jointly by FZI, the University of Karlsruhe, and the University of Manchester. It is available freely for scientific and academic purposes. It implements the OWL-DL standard without nominals (which is not needed within the tasks described here). It features SWRL reasoning with so called DL-safe rules, which is sufficient for the given tasks.
- *Pellet* is developed by the MINDSWAP group of the University of Maryland. It accompanies the SWOOP editor, and, although tightly integrated into it, can also be used as a standalone reasoner. It implements the full OWL-DL standard and also the DL-safe set of SWRL rules. Although the Pellet reasoner comes with a warning that it is not optimised for speed, the ROVE ontology does seem to be small enough to support interactive use with reasonable response times.
- *Hoolet* is an implementation of an OWL-DL reasoner developed by the University of Manchester that uses a first order prover. It consists of a graphical front end that allows loading of ontologies and rule sets, along with a reasoner. The prototype provides a useful tool but only for small examples and is for Linux only.
- *Jess* is a rule engine. Golbreich and Atsutoshi (2004) describes how JESS and a DIG enabled reasoner can be used together in order to reason over rule enhanced OWL-DL. Thus the shortcoming of the current DIG interface can be avoided.

5 Possible solution

After the short introduction, brief deception of some editors and reasoners, and description of the requirements and outcomes of the exercise it is time for the presentation of a possible answer to the given problem. Nevertheless before searching for the solution to the task the problem should be split into small subproblems which allow the students to better understand the approach, its goals and the final solution. In particular it means that the students should recognise the limitation of OWLs abilities and the capabilities provided by adding rules. Due to this goal, in the first step the students will build an ontology which describes the group issue as far and detailed as possible. In the second step, the open questions which could not be covered by the ontology are to be defined using rules.

5.1 Building an ontology

The ontology requires the representation of each student and tutor taking part in the miniproject activity, the groups themselves, and attributes including:

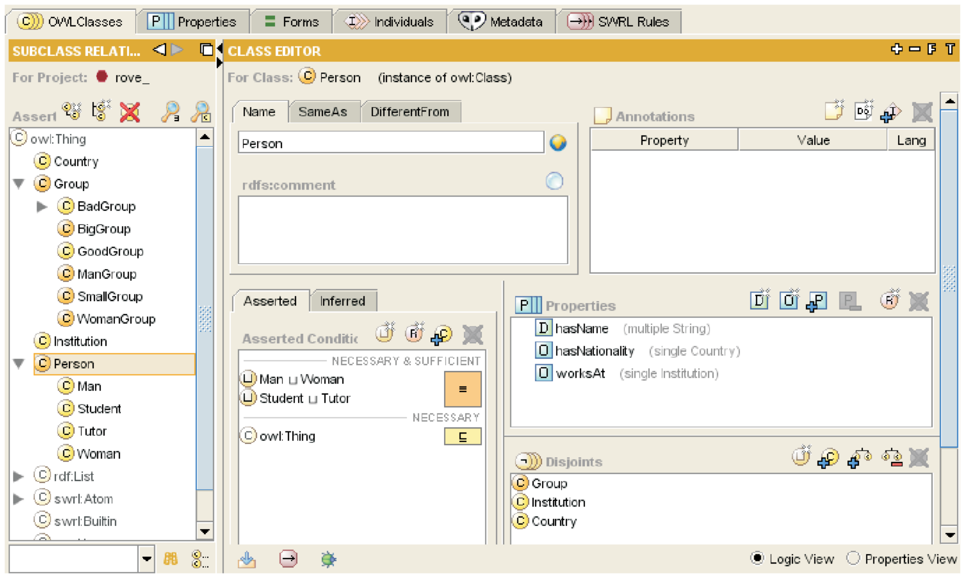
- for persons: nationality, associated institution, and gender
- for groups: group membership (of students) and group leadership (of tutors).

Some of these terms can be defined as *disjoint concepts* within an ontology which should describe the domain of summer school taking into account the conditions described in Section 3: `Person`, `Country`, `Institution`, and `Group` (see Figure 2). Furthermore, `Person` is divided, on the one hand, into disjoint subclasses `Tutor` and `Student` partitioning the class completely and on the other hand by gender, into disjoint subclasses `Man` and `Woman`, also as a complete partition.

To describe a person in the context of the summer school the person's name, nationality, and the corresponding institution, at which the person works, needs to be defined by building (object and data) properties `hasName`, `hasNationality` and `worksAt`, respectively. Furthermore, each student is a member of exactly one group (`memberOf`) and has exactly one favourite tutor (`attractedTo`). Additionally, the summer school project groups have names (`hasLongName`, `hasShortName`), are led by a tutor (`ledBy`) and have members (`hasMember`) only from the class `Student`.

To find the limitations of OWL, the students should first try to formalise and implement as many requirements (cf. Section 3) as possible using the developed ontology and staying in the framework of the OWL-DL.

Figure 2 ROVE-ontology (see online version for colours)



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

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

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

With the ROVE data, all the groups satisfy this condition and cardinality conditions are easily formulated within OWL.

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

As seen above, to satisfy Condition #1 the relationship between a `Group` and each `Student` may be represented with the property `hasMember` (inverse to `memberOf`) and inserted cardinality restrictions. This method can not be applied to the Condition #2 since it requires `hasMember` to have a minimum cardinality for values from each of the subclasses `Man` and `Woman` (so called *qualified* cardinality restrictions, which are not available in OWLDL, but are included in the proposal for OWL 1.1 (Cuenca Grau et al., 2006)). To cope with this problem one can use 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, it needs to be captured that this condition was only *one* of those to be satisfied by a good group (cf. Section 3.2), so it must not be a sufficient condition for a good group. To face this issue, one could approach the problem in reverse way by specifying conditions for being a bad group (`BadGroup`) rather than a good group[4].

In order to state a sufficient condition[5] for a class `BadGroup` a class `ManGroup`, which is a group with male members only, must be introduced:

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

The same applies for `WomanGroup`:

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

The class `BadGroup` is defined as a subclass of `Group`. Now it is simple to implement these two sufficient conditions for being a ‘bad group’ by making `WomanGroup` and `ManGroup` subclasses of `BadGroup`. This means that every `ManGroup` and every `WomanGroup` is also necessarily a `BadGroup`, thus rendering the necessary and sufficient conditions of the subclass sufficient conditions of the superclass.

$$WomanGroup \sqsubseteq BadGroup$$

$$ManGroup \sqsubseteq BadGroup$$

However, having set this up, the automatic classification will not classify one of the groups as a `BadGroup` with the ROVE data, although it consists of four male students. This is 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`. There are several ways to handle this problem: one could define groups by enumeration (but not the restrictions many reasoners have regarding nominals (Hladik, 2003)) or the size of the groups could be stated explicitly by creating two new concepts: `BigGroup` (group with exactly five members) and a `SmallGroup` (group with exactly four members) as disjoint subclasses of Group (other conditions will be asserted 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 require 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 is not possible to formulate them using only OWL.

From the five requirements defined in Section 3 only two of them can be implemented using OWL. Here we summarise the experiences the students should have by now:

- the requirement on each group to have 4 or 5 members can be expressed using cardinality on the `hasMember` property for groups
- sufficient conditions for a `BadGroup` can be stated by defining the additional concepts `WomanGroup` and `ManGroup`
- the requirements #3–#5 are not expressible in OWL, as they need property chaining.

5.2 Defining rules

As not all requirements can be implemented using OWL alone, in the next step the students investigate some solutions using rules. For this purpose the usage of the SWRL (Horrocks et al., 2003) (available with a plug-in to Protégé, or natively in SWOOP) within the ROVE ontology described above will be explored.

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.* SWRL provides Horn-like rules for both OWL-DL and OWL Lite, includes a high-level syntax for representing these rules (Aggarwal, 2004) and is more powerful than either OWL-DL or Horn rules alone (Horrocks et al., 2005). In SWRL one may assert equivalences as well as implications.

The first step towards the application of the SWRL to the group scenario is the definition of the conditions 3 and 4 which reflect that mini-project groups should have members of all different nationalities, and be from different institutions, i.e., no two members of the group should have the same nationality, or be from the same institution.

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

At this point it would be intuitive to come up with a definition for a group in which all members are from different nationalities – `InternationalGroup` (which, in turn, would be another necessary condition for a `GoodGroup`). Nevertheless, a correct definition of such a group will be a very challenging task since one has to define different rules for different types of groups: one rule for groups with four members, and another for those groups which have five members. This results in a high number of necessary comparisons between the nationalities of the group members. Since every pairwise combination has to be considered, the rule for four member groups takes into account six different comparisons, whilst the five member groups needs ten (it is $\sum_{n}^{n-1} n$ comparisons for n members, thus growing quadratically). To overcome this problem, instead of detection good groups, the students should construct a rule which determines bad groups. This means, a rule that states if a group has any two members with the same nationality it is a `BadGroup`, is needed (‘SameNationalitiesRule’):

- *Natural language*: if the group has two members with the same nationality it is a *BadGroup.*
- *Explanation of SWRL notation with natural language*: If group `g` has member m_1 and member m_1 has nationality `n` and group `g` has another member m_2 who is different from member m_1 and member m_2 also has nationality `n` (member m_1 and member m_2 have the same nationality `n`) then the group `g` is a `BadGroup`.
- *SWRL notation*:

  $hasMember(?g, ?m_1) \wedge hasNationality(?m_1, ?n) \wedge$

  $hasMember(?g, ?m_2) \wedge hasNationality(?m_2, ?n) \wedge$

  $differentFrom(?m_1, ?m_2) \rightarrow BadGroup(?g)$

Condition #4: Groups should have members from all different institutions, that is, no two members of the group should be from the same institution

The same approach applies to different institution as was used for different nationalities:

$hasMember(?g, ?m_1) \wedge worksAt(?m_1, ?n) \wedge$

$hasMember(?g, ?m_2) \wedge worksAt(?m_2, ?n) \wedge$

$differentFrom(?m_1, ?m_2) \rightarrow BadGroup(?g)$

Condition #5: Groups should have fun

The criteria for a ‘fun group’ was chosen in order to learn more about OWL’s and SWRL’s abilities to represent and reason with compositions of properties and situations where there were multiple properties that connected the classes. The definition is: a fun group is one where all the students in the group are attracted to the tutor leading their group. One further motive here is to provide some amusement for the students.

The following rule provides a formalisation of the criteria:

$$hasMember(?g, ?m) \wedge attractedTo(?m, ?t_1) \wedge hasTutor(?g, ?t_2) \wedge differentFrom(?t_1, ?t_2) \rightarrow BadGroup(?x)$$

which captures that any group that has a member who is attracted to someone other than the group's tutor is a bad group.

5.3 *OWL: not bad does not equal good*

At this point the students may think they have captured all requirements adequately using OWL and rules, as they are now able to automatically classify all bad groups. However, it is not possible to classify good groups yet!

Stating that good groups are all groups that are not bad groups relies on 'negation as failure' which is not supported by OWL. Thus whilst automatic classification may be used to identify bad groups, it is not able to identify good groups simply on the grounds of not being bad. 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 for bad groups is exhaustive.

In order to classify `GoodGroups` as such, one needs to reformulate all the prior requirements as 'positive rules', defining sufficient conditions for classification as a `GoodGroup`. The positive form of the rule often becomes long and unwieldy, as discussed with the `InternationalGroup`. The same applies for `InterInstitutionalGroup`[6], having another huge rule with more than 20 conjunctions in its body. Also a rule to define a `FunGroup` is needed. Although the intuition behind the definition of a `FunGroup` is easy – a group where its members all were attracted to the very tutor leading the group – formalising the rule again is an extremely tedious task, again leading to a huge and hard to maintain rule.

Many students will find it particularly challenging to fully understand the implications of the asymmetric structure of the positive and negative forms of rules, as intuitively they are only attempting to view the situation from the opposite side.

On the other hand, a `MixedGenderGroup` is much more easily described (actually, this is possible in pure OWL again):

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

Finally, having formalised the requirements as rules and being able to classify whether a given group passed each rule, a `GoodGroup` can be defined as the intersection of all groups meeting each single requirement. Here an OWL axiom can be created, using class descriptors actually defined in the SWRL part of the ontology.

$$GoodGroup \equiv MixedGenderGroup \sqcap InternationalGroup \sqcap$$

$$InterInstitutionalGroup \sqcap FunGroup$$

6 Problems

In Section 4, some technical problems that may be encountered with the usage of the current tools have been already mentioned. Besides such problems – that will make the students aware of the current state of the art in Semantic Web technology – they may also run into issues like performance problems with the reasoners. This will allow the teacher to explain why some constructs or combination of constructs show undesirable properties for reasoning, and how they can be avoided.

Besides these practical problems, this chapter points out an issue students have faced when understanding the conceptualisation and semantics of the ROVE ontology, and trying to achieve the goals outlined in Section 3. The main problem for most students seems to be to understand the *Open World Assumption* and its consequences. This is easily demonstrated while solving Condition #2 above: although the student has already defined that a `ManGroup` is a group where all members are male, there are still male groups (with only male members) which were not classified as a `ManGroup`. This happened due to the fact that if we consider a group with four members without explicit specification that this group do not have the fifth member the reasoner assumes that the hypothetical fifth member could be female. This result is usually a surprise to the students, who by this point are likely to believe they have adequately covered all the stated conditions. In our experience, the struggling to understand the needed formalisation and capture the sufficient conditions for `GoodGroup` turns out to be a great chance for focussed discussions and real insights into how Description Logics work. As we experienced ourselves, the tutor can easily help with a few well placed hints, and still allow the students to come to most of the solution themselves, thus deepening the learning experience.

7 Resources for teachers and students

This section offers a list of links with further resources regarding the topics covered in this paper. Instead of typing these links by hand, you can also go to the ROVE homepage on http://km.aifb.uni-karlsruhe.de/projects/rove and find an up to date electronic list of the given resources.

7.1 Reasoners

Hoolet: implementation of an OWL-DL reasoner that uses a first order prover

- download: http://owl.man.ac.uk/hoolet/

RACER/RacerPro: the first OWL Reasoner

- documentation: http://www.sts.tu-harburg.de/~r.f.moeller/ racer/
- download: http://www.racer-systems.com/de/index.phtml? lang

KAON 2: an OWL reasoner

- download: http://kaon2.semanticweb.org/
- publications: http://www.aifb.uni-karlsruhe.de/Publikationen/ showPublikationenProjekt?id_db=62
- command line tools to work with owl ontologies: http://owltools.ontoware. org

Pellet: an open-source Java based OWL-DL reasoner

- documentation: http://www.mindswap.org/2003/pellet/
- download: http://www.mindswap.org/2003/pellet/download.shtml

FaCT++: OWL-DL reasoner

- download: http://owl.man.ac.uk/factplusplus/

7.2 Ontology editors

Protégé: an ontology editor and a knowledge-base editors:

- documentation: http://protege.stanford.edu/
- download: http://protege.stanford.edu/plugins/owl/index.html
- tutorial: http://www.co-ode.org/resources/tutorials/ProtegeOWLTutorial.pdf

Swoop: a Hypermedia-based Featherweight OWL Ontology Editor

- download: http://www.mindswap.org/2004/SWOOP/

7.3 Language specifications

OWL: Web Ontology Language

- documentation: http://www.w3.org/TR/owl-features/
- http://www.w3schools.com/rdf/rdf _owl.asp
- http://www.cs.vu.nl/~frankh/postscript/OntoHandbook03OWL.pdf
- experiences with teaching OWL (Rector et al., 2004).

SWRL: Semantic Web Rules Language

- current proposal: http://www.w3.org/Submission/SWRL/
- XML cover pages on SWRL: http://xml.coverpages.org/ni2004-05-21-a.html

8 Teaching guidelines

8.1 Teaching items

The case study can be used as supporting material in the undergraduate teaching of the basics within the Semantic Web field. The study highlights a practical situation considering the interaction between an ontology defined in the Web Ontology Language (OWL) and rules as well as the issue of the Open World Assumption. Since it is an easy to understand scenario with clear defined requirements the teaching case can be used for teaching both groups of students: students that are not familiar with Semantic Web technologies (whereas some basic knowledge and experience in building ontologies would be very helpful) as well as during the lessons with students with some previous knowledge in this field. Depending of the level of knowledge we would recommend different duration/amount of the lesson items:

- *Students with no experiences*. As the students have no previous knowledge about building ontologies a breakdown of the case into two subproblems is recommended:
  - In the first teaching item the students should concentrate on building an ontology which satisfies the first two requirements (Conditions #1 and #2). They should learn about the concepts, data and object properties, individuals, etc. and work with ontology editors like Protégé.
  - In the second item, since the students have already built/worked with an ontology, the notion of rules and their development should be introduced. At this point the students can start to analyse the remaining conditions (Conditions #3–#5) and trying to formalise them using at first only OWL-DL and then the rules (in particular SRWL). The main issue within this item, apart from the rules, is the Open World Assumption.
- *Students with some experiences*. If the case study is used to work with students that have already some basic (general) knowledge about ontologies or even some experiences in developing own ontologies they can, from the beginning on, analyse all given requirements. This means the two abovementioned teaching items can be integrated into one.

After two or one teaching item respectively, the students should be able to work with OWL ontologies, define rules (in SWRL) and be able to explain the problem of the Open *World Assumption*. If the time allows, it is recommended to let the students formalise their own class, instead of taking the summer school ontology for the ground data. It is usually more fun to work on data the students can relate to. The fastest way to approach this, is to allow each student, or group of student, to formalise their own data, and then merge the data.

8.2 System requirements

Most of the tools are written in Java and can thus run on several different operating systems. It is suggested to test the chosen applications with the local installation of Java, as some of the suggested tools require Java 5.

Currently available Semantic Web tools have usually been developed within research institutions, and have not been not optimised with regards to performance. Reasoners especially may make heavy claims on both memory and processor time. Also some advanced features of the ontology development environments, like the explanation module shown in Figure 1 may require considerable resources. For these reasons, the system requirements and response times will depend heavily on the chosen tools, and the size of the final ontology. Our experience with these exercises has shown that most commonly available hardware can deal adequately with ontologies of the size used in the examples presented here (i.e., less than 100 instances and only a small number of classes and properties). However, it is recommended that the teacher run some preliminary tests using the local hardware and chosen tools prior to running tutorials, to ensure that the tools will work in the local environment with reasonable response times.

9 Conclusion

This paper gave a brief overview of a Semantic Web teaching case presenting, by means of an easy-to-understand example, some limitations of OWL-DL and first steps in using SWRL rules. To facilitate practical exercises some currently available ontology editors and additional Semantic Web tools, which may be used to construct and work with ontologies and rules to complete the described task, were described. The experiences with ontology development and the usage of different tools enabled the authors to highlight some commonly-experienced problems and limitations of the available technologies. In the context of the easy summer school scenario the difficult issue of the 'Open World Assumption' was pointed in a way which should be readily accessible to students and may encourage their interest in the underlying formalism.

In conclusion, the authors hopes that this teaching case will serve to assist 'young players' and their guides in the practical navigation of Semantic Web Technologies whilst avoiding some common traps and pitfalls.

Acknowledgements

We would like to thank Antoine Zimmermann, who was a member of the original ROVE mini-project team at SSSW2005; the organisers of the SSSW2005: Enrico Motta and Asuncion Gomez-Perez; and all the students, tutors and speakers who took part. Research reported in this paper has been partially financed by NICTA (http://www. nicta.com.au), the EU project SEKT (http://www.sekt-project.com), the NoE KnowledgeWeb (http://knowledgeweb.semanticweb.org), and the Ger- man Ministry of Research (BMBF) project Knowledge Nets (http://wissensnetze. ag-nbi.de) which is a part of the InterVal (http://interval.hu-berlin.de) Berlin Research Centre for the Internet Economy (http://www.internetoekonomie. net).

References

Aggarwal, R. (2004) *Semantic Web Services Languages and Technologies: Comparison and Discussion*, LSDIS Lab, University of Georgia, http://citeseer.ist.psu.edu/700682.html

Bechhofer, S., Horrocks, I., Patel-Schneider, P.F. and Tessaris, S. (1999) 'A proposal for a description logic interface', *Proc. of the Description Logic Workshop (DL'99)*, pp.33–36, *CEUR Workshop Proceedings*, http://ceur-ws.org/Vol-22/

Bechhofer, S., Liebig, T., Luther, M., Noppens, O., Patel-Schneider, P.F., Suntisrivaraporn, B., Turhan, A-Y. and Weithöner, T. (2006) 'DIG 2.0 – towards a flexible interface for description logic reasoners', in Cuenca Grau, B., Hitzler, P., Shankey, C. and Wallace, E. (Eds.): *Proc. of the 2nd Workshop OWL Experiences and Directions 2006.*

Cregan, A., Mochol, M., Vrandecic, D. and Bechhofer, S. (2005) 'Pushing the limits of OWL, rules and Protege – a simple example', in Cuenca Grau, B., Horrocks, I., Parsia, B. and Patel-Schneider, P. (Eds.): *OWL: Experiences and Directions*, Galway, Ireland.

Cuenca Grau, B. *et al*. (2006) T*he OWL 1.1 extension to the W3C OWL Web Ontology Language*, University of Manchester, http://webont.org/owl/1.1/index.html

Golbreich, C. and Atsutoshi, I. (2004) 'Combining SWRL rules and OWL ontologies with Protege OWL plugin, Jess, and Racer', in Fergerson, R. and Noy, N. (Eds.): *Proceedings of the 7th International Protege Conference*, Bethesda, Maryland, USA, http://protege.stanford.edu/conference/2004/abstracts/Golbreich.pdf

Hladik, J. (2003) 'Reasoning about nominals with FaCT and RACER', *Proc. of the 2003 International Workshop on Description Logics (DL2003)*, CEUR-WS, http://SunSITE.Informatik.RWTH-Aachen.de/Publications/CEUR-WS/Vol-81/hladik.pdf

Horrocks, I., Patel-Schneider, P.F., Bechhofer, S. and Tsarkov, D. (2005) 'OWL rules: a proposal and prototype implementation', Conditionally accepted for publication', *Journal of Web Semantics*, Vol. 3, No. 1, pp.23–40.

Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2003) 'SWRL: a semantic web rule language combining OWL and RuleML', *DARPA DAML Program*, http: //www.w3.org/Submission/2004/SUBM-SWRL-20040521/

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H. and Wroe, C. (2004) 'OWL pizzas: practical experience of teaching OWL-DL: common errors & common patterns', in Motta, E., Shadbolt, N.R. and Stutt, A. (Eds.): *Proc. of the 14th EKAW*, Springer, Vol. 3257 of LNCS, Whittlebury Hall, UK, pp.63–81.

Notes

[1]http://www.w3.org/TR/owl-features

[2]http://www.w3.org/Submission/SWRL

[3]Check the ROVE website http://km.aifb.uni-karlsruhe.de/projects/rove for links to tools and other materials.

[4]Note that not satisfying any one of the five conditions is sufficient for classification as a bad group.

[5]Note that Protégé supports explicit specification of either '*Necessary*' or '*Necessary and Sufficient*' asserted conditions through the interface but no conditions which are '*Sufficient*' without being '*Necessary*'.

[6]InterInstitutionalGroup – a group where all members are from different institutions.