Exercise 8 Group 14

Jon Volkmar
Heiko Baumann
Peter Hirschfeld

1. Please describe in your own words the difference between RDFS amd OWL, and then between OWL Lite, OWL DL, OWL Full and OWL 2. Create some simple examples to illustrate the difference.

   Answer: OWL is more expressive than RDFS: with OWL we can define relations between classes, constraints and cardinalities, equivalences of classes and properties, create properties of properties, use boolean combinations and other expressive constructs (e.g. disjointnes, inverse properties, property restrictions, enumerations). Also, special properties can be expressed, namely transitive, symmetric, functional, and inverse functional properties. However, some RDF markup is preserved in OWL, and instances of classes are declared in RDF. An example of a transitive property that cannot be expressed in RDFS:

   ```
   <owl:ObjectProperty rdf:ID="subRegionOf">
     <rdf:type rdf:resource="&owl;TransitiveProperty"/>
     <rdfs:domain rdf:resource="#Region"/>
     <rdfs:range  rdf:resource="#Region"/>
   </owl:ObjectProperty>
   ```

   OWL Lite is a simplified subset of OWL for tool developers, with some features from OWL, supporting class hierarchies and constraints. However, many features and constructs from OWL are not found in OWL Lite, such as owl:disjointWith.

   OWL DL: DL stands for description logic. It is restricted compared to OWL Full in that classes cannot be simultaneously be properties and individuals, and properties cannot simultaneouly be individuals. This is called vocabulary partitioning. As well, the set of object properties and data type properties are disjoint in DL. DL also requires explicit typing. No transitive cardinality constraints can exist in OWL DL. Another difference is that there are some restrictions placed on the usage of anonymous classes.

   OWL 2 has several new constructs, such as qualified cardinality restrictions, role chains, and expressive data predicates. It has some different syntaxes, and targets the XML technology toolchain with OWL/XML, which is a new non-RDF XML syntax. It has added features for supporting datatypes, metamodelling, annotation, and database style keys. Property chains are also introduced: for example saying that **{ ?x ex:uncle**

?y } is equivalent to **{ ?x ex:parent ?z . ?z ex:brother ?y . }** can be expressed as a property chain.

2. Ontology modelling application scenario

   (a) Prepare a brief project documentation describing your work on the tasks b - f. Your documentation should contain the following information:

   - Time spent on solving each task (person hours).
     - (b) 30 minutes for each person
     - (c) 30 minutes each
     - (d) 30 minutes each
     - (e) 1 hour each
     - (f) 2 hours each
   - Time spent discussing each task with your teammate(s).
     - (b) 10 minutes
     - (c) 10 minutes
     - (d) 30 minutes
     - (e) 30 minutes
     - (f) 20 minutes
   - Difficulties encountered.
     - i. When making the ontology in Protege, the software would sometimes crash when we tried to run the reasoner. And, if it didn't crash, it would sometimes yield unexpected results. Namely, an individual from class A, defined as disjoint from anotherclass B, would be inferred to be from class B anyway.
   - If you encountered difficulties: How did you solve them? How long did it take you to solve the difficulties (person hours)? Difficulties could be technical issues with the software used, understandability of the assignment, communication issues with your teammates, etc...
     - i. Eventually we abandoned the reasoner in Protege and stopped creating individuals, as everything else appeared to work as expected, and the reasonser is not necessary for the assignment. This took about an hour.
     - ii. It was difficult to figure out a way to model different class sizes, allowing each class to have a different maximum size but without making a new subclass every time. It was thought about and we settled on a compromise, creating a few subclasses representing different sizes (10, 50, and 100 students) which can be used for any number of courses. This took about half an hour.
   - If you needed to apply changes (e.g. while building your ontology you realized that some competency questions do not work in your scenario, and you needed to rework your competency questions), document that.

(b) Competency questions: Write down at least 20 competency questions which the intended application should be able to answer on the basis of the ontology.

Answer:

i. For a given course, who is the lecturer or tutor?
ii. How many students are enrolled in a given class?
iii. Where and when is a lecture given?
iv. What are the names of the courses a given student is enrolled in?
v. Which lectures occur on a given day of the week?
vi. What are the names of the students enrolled for a given course?
vii. How many times does a course occur per week?
viii. How many instructors are there for a given course?
ix. Who is allowed to loan a given faculty resource (e.g. room, projector, PC...)
x. For a given room, what dates is it booked for?
xi. What is the maximum loan period for a given faculty resource?
xii. Is a resource currently on loan?
xiii. What is the capacity of a given course?
xiv. Is a given course a lecture or a seminar?
xv. What prior courses are required for a student to participate in a given courses?
xvi. Which courses is a student eligible to enroll for?
xvii. For a given student, which courses need to be completed before they can enroll in a certain course?
xviii. Which resources is a given person allowed to loan?
xix. Is a particular resource loanable?
xx. What are the names of all the students enrolled in a course which takes place on a Monday?

(c) Glossary: Extract key terms from the competency questions which frequently occur and are of importance for the given application domain.

Answer:

- Course
- Lecturer
- Tutor
- Student
- Course Name
- Course capacity
- Loan period
- Booked dates
- Lecture
- Seminar
- Required courses

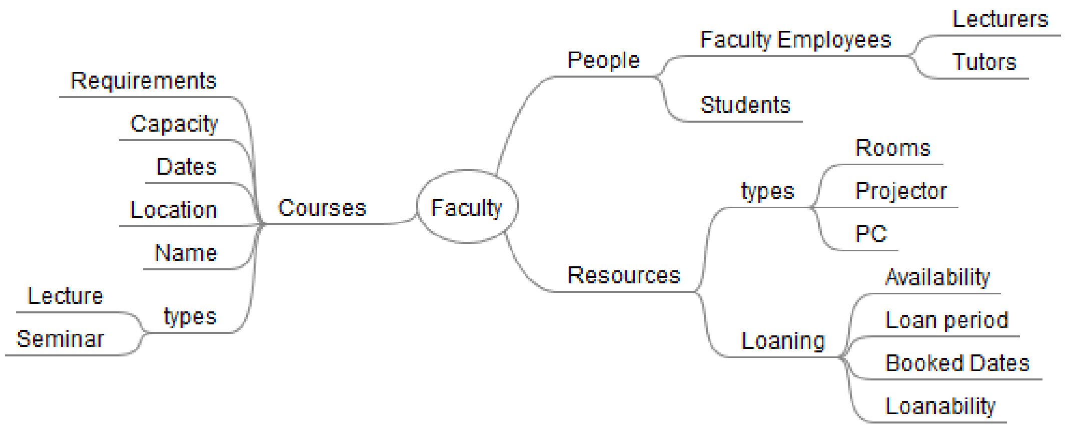
- Loanability

(d) Mindmap: Create mindmaps to model the relationships between the key terms of your glossary in a light-weight way.

Answer:

Mindmap created with Freemind:

Figure 1: Mindmap of the "Faculty" ontology

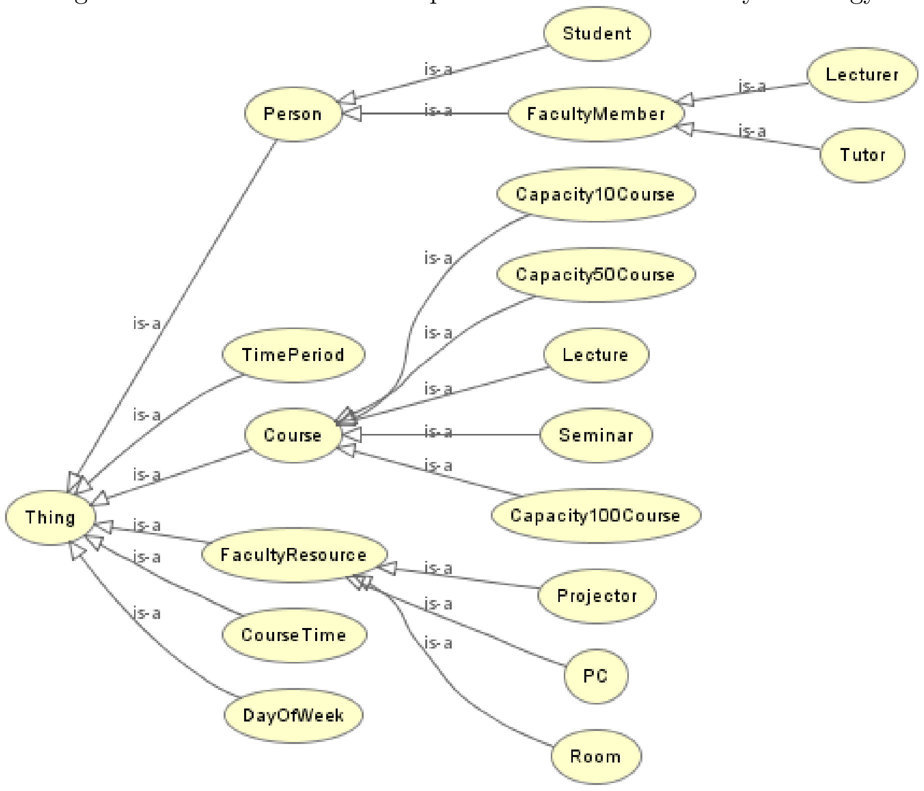


(e) Ontology engineering: Derive from the mindmaps an ontology containing some first classes. Use a ontology editor for this task, e.g., Protg(http://protege.stanford.edu/). Add labels in English and German language to each concept of your ontology

Answer:

Ontology created with Protege. Please see ex8.owl for the OWL/XML markup. The following diagram, generated by the OwlViz tool in Protege, shows the classes of the ontology. It does not show the object and data properties which are used in restrictions fufilling our scenario requirements, to view these the owl file must be opened in Protege or simply in a text editor.

Figure 2: Classes in the OWL representation of the "Faculty" ontology



(f) Go to the Watson Website (http://watson.kmi.open.ac.uk/WatsonWUI/) and search for existing ontologies modeling the same or similar concepts as you have in your ontology. Refine your ontology by reusing concepts and relationships of the found ontologies.

We were not able to find any existing ontology close enough to our own to be easily adaptable. However, maybe it would be possible to model "temporal events" such as meetings and generic time intervals, as done in (http://daml.umbc.edu/ontologies/cobra/0.4/academia).