Semantic Web - KEN3140

Lab 8: 60min

Constructing advanced OWL axioms in Protege, performing advanced reasoning tasks and ontology debugging

Kody Moodley & Michel Dumontier, 22 September 2021

Purpose of this document

This document outlines your tasks for Lab 8 which follow on from Lab 7. You will load a variation of the family relations ontology from Lab 7 into Protégé and extend the ontology to describe more conceptual knowledge about family relations, which special focus on the constructs and reasoning assumptions covered in Lecture 8. Required learning materials: Lecture 7 and 8.

Learning objectives:

1. How to identify which kinds of knowledge cannot be captured in ALC and require additional constructs from SROIQ (OWL 2 DL)
2. How to identify the specific DL features required to capture a logical statement about the domain
3. To be able to anticipate the effects on reasoning when enforcing constraints such as transitivity and symmetry to roles (for smaller ontologies with less than 15 individuals)
4. To be able to explain the intuitions behind Open World Assumption and Unique Name Assumption and to anticipate the effects they have on entailments for smaller ontologies
5. How to use the OWL reasoner and explanation facilities in Protege to diagnose logical incoherences in an ontology
6. How to use explanations (justifications) for logical incoherences to pinpoint problematic axioms and formulate strategies to rectify the errors

Tasks

Task 1 [20min]

In Task 1 of Lab 7B you added equivalence axioms to define classes such as Grandmother, Aunt, Cousin etc. Now, in this task, your objective is to:

- Pick 3 - 5 individuals from the ABox who are persons that have children, parents **and** siblings. Run the reasoner (HermiT) to see what inferences you can observe about these

individuals. Remember the inferences (implicit axioms) are highlighted with a yellow background in Protege.

- Now, identify new *relations* to add into the *object property* hierarchy for **father, mother, grandfather, grandmother, relative, son and daughter**. **NB:** you have to *reuse existing vocabulary* - do not create your own custom relations. Possible sources to look for vocabulary are: schema.org, wikidata.org, linked open vocabularies, foaf and dbpedia, CWRC (ontology of the Canadian Writing Research Collaboratory).
- Make a choice about where to place these new properties in the property hierarchy in Protege and add them into the hierarchy.
- For the 3-5 individuals you identified in the first step of this task, remove *only* those property assertions concerning parents, children and siblings for these individuals. Replace these property assertions with new ones using the relations **father, mother, son and daughter** instead, which you introduced in the previous steps.
- Run the reasoner (HermiT) again. Observe the inferences made and whether your changes to the ontology have changed the inferences that the reasoner makes. Did your changes influence the inferences made? If so, how?

Task 2 [20min]

Once you have added the relations from Task 1 in the correct locations in the hierarchy:

- Select which role constraints (transitivity, symmetry etc.) to enforce on these. You can toggle the checkboxes for these constraints in the object properties tab of Protege. Verify the effect that these constraints have on the inferences made by the reasoner, by observing the inferences made before toggling the constraints, and then after. **Note:** transitivity and asymmetry cannot both be used together on a property (this is no longer part of OWL, although Protege allows you to state this)
- Create two unique **role composition axioms** to capture different ways that someone can be a **relative** to another person in the ontology. To give an example of one for inspiration: we know that if a is a **parent** of b and b is a **sibling** of c then a is a relative of c. Create a further two unique **role composition axioms** to capture different ways that someone can be inferred to be a **grandparent** to another person. Verify the effect that these constraints have on the inferences made by the reasoner. To do this, you have to observe the inferences of the reasoner *before* you add your axioms, and then again *after* you add them.

Task 3 [20min]

Your first task is to:

- Identify all members of the family in the ontology who have at least 2 sons. **Hint:** you will use the DL Query tab to express a class expression query which you can then execute to return your results (and the answer). Check the Lecture 7 and 8 slides to see what DL feature you need to express this class. If you execute your query without modifying the

ontology, you may notice that it will not return any results. **Hint:** you need to tell the OWL reasoner that all the individual names cannot refer to the same objects in the domain (we want to simulate the unique name assumption - UNA) - see Figure 1.

Figure 1: UI component potentially useful for Task 3.

After modifying the ontology to make sure the UNA is being made, execute the query again in the DL query tab to see if there is a difference in the results.

- Examine the definitions for “Nerd” and “SportsFanatic” in the ontology. Modify the definition of “Nerd” to mean that:
    - Only those people having **all** their hobbies as either “Cosplay”, “PCgame” or “Comic” (or any combination thereof) **with no other hobbies,** are considered as a “Nerd”.
    - Run the reasoner before and after to see who is classified under “Nerd” and compare the results.
- Modify the definition of “SportsFanatic” to mean that only those people that have at least 3 unique hobbies that concern sports are considered a “SportsFanatic”. Do you get different instances of “SportsFanatic” after changing the definition?
  Hint: this task may be trickier than the previous one. In addition to changing the definition of “SportsFanatic”, you will need to use *disjoint classes* to get the answer you are looking for. Run the reasoner before and after to see who is classified under “SportsFanatic” and compare the results.

Contact:
Kody Moodley (kody.moodley@maastrichtuniversity.nl)
Michel Dumontier (michel.dumontier@maastrichtuniversity.nl)