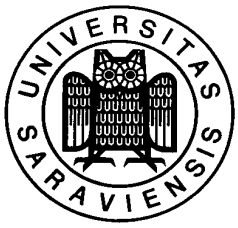
Universität des Saarlandes

Fakultät 6 – Fachrichtung 6.2 - Informatik

Prof. Dr. Jörg Hoffmann
Prof. Dr. Wolfgang Wahlster



Foundations of Artificial Intelligence, SS 2016

Assignment 10 – Solution

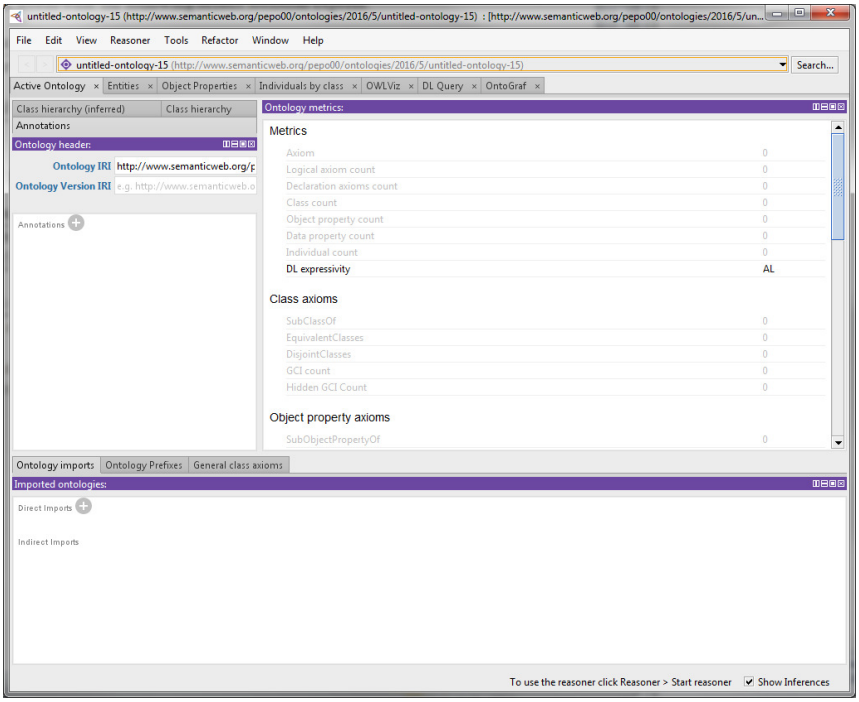
Exercise 1: [OWL - Preparation] (0 Points)

1) Install and Launch Protégé 5.0 as described on http://protegewiki.stanford.edu/wiki/Install_Protege5

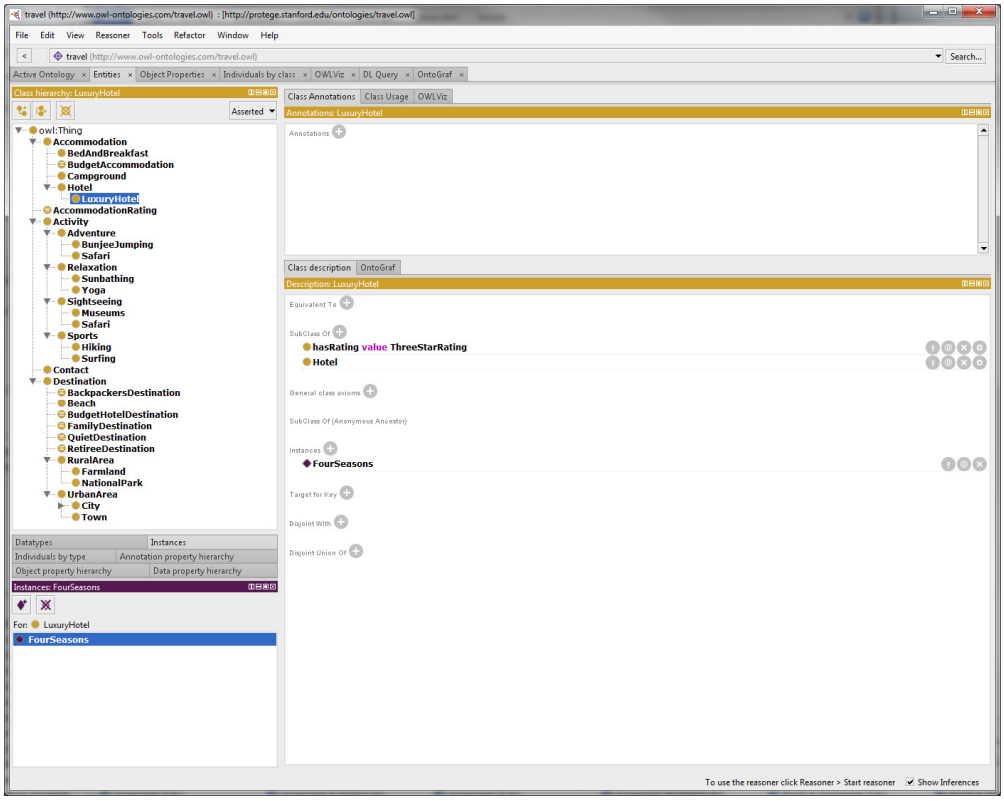
2) Load the travel ontology known from the lecture into Protégé 5.0 using "File -> Open from URL... -> http://protege.stanford.edu/ontologies/travel.owl" and get a bit familiar with the ontology using the different Protégé 5.0 views (see also, e.g., the documentation given on http://protegewiki.stanford.edu/wiki/Protege4Views)

Sample Solution:

1)



2)



Points: no points

Exercise 2: [OWL – Ontology Browsing] (4 Points)

1) Browsing classes: (0.5 points)

One class definition is wrong. Which one and why?

2) Browsing instances: (1 point)

a) There is one *LuxuryHotel* instance in the ontology. What is its name and where is it located? (0.5 points)

b) How many instances are in the ontology? Which ones? (0.5 points)

3) Browsing class properties (relations): (1 point)

a) Which relations are in the travel ontology? Write them down incl. domain and range. (0.5 points)

b) Which of them are inverse to each other? (0.5 points)

4) Browsing data properties: (1 point)

Which data properties are in the travel ontology? Write them down incl. domain and range.

5) Running the classifier: (0.5 points)

The classifier (via "Reasoner -> Start Reasoner") infers that a *NationalPark* is a *BackpackersDestination* as shown in the lecture. Two more inferences are drawn for the travel ontology. Which ones?

Sample Solution:

1) *Safari* is not properly specified because it is realized as a subclass of two different classes at the same time, namely *Adventure* and *Sightseeing,* which is not allowed. (0.5 points)
2) a) *FourSeasons* is an instance of *LuxuryHotel* and according to the ontology it is located in *Sydney*. (0.5 points)

   b) There are all in all 14 instances in the ontology. (0.5 points)
   (3 instances of *AccommodationRating: OneStarRating, TwoStarRating, ThreeStarRating*,
   2 instances of *Beach: CurrawongBeach, BlondiBeach*,
   2 instances of *Capital: Sydney, Canberra*,
   1 instance of *City: Cairns*,
   1 instance of *LuxuryHotel: FourSeasons*,
   2 instances of *NationalPark: Warrumbungles, BlueMountains*,
   2 instances of *RuralArea: CapeYork, Woomera*,
   1 instance of *Town: Coonabarabran*)
3) a) There are 6 class properties in the ontology: (0.5 points)

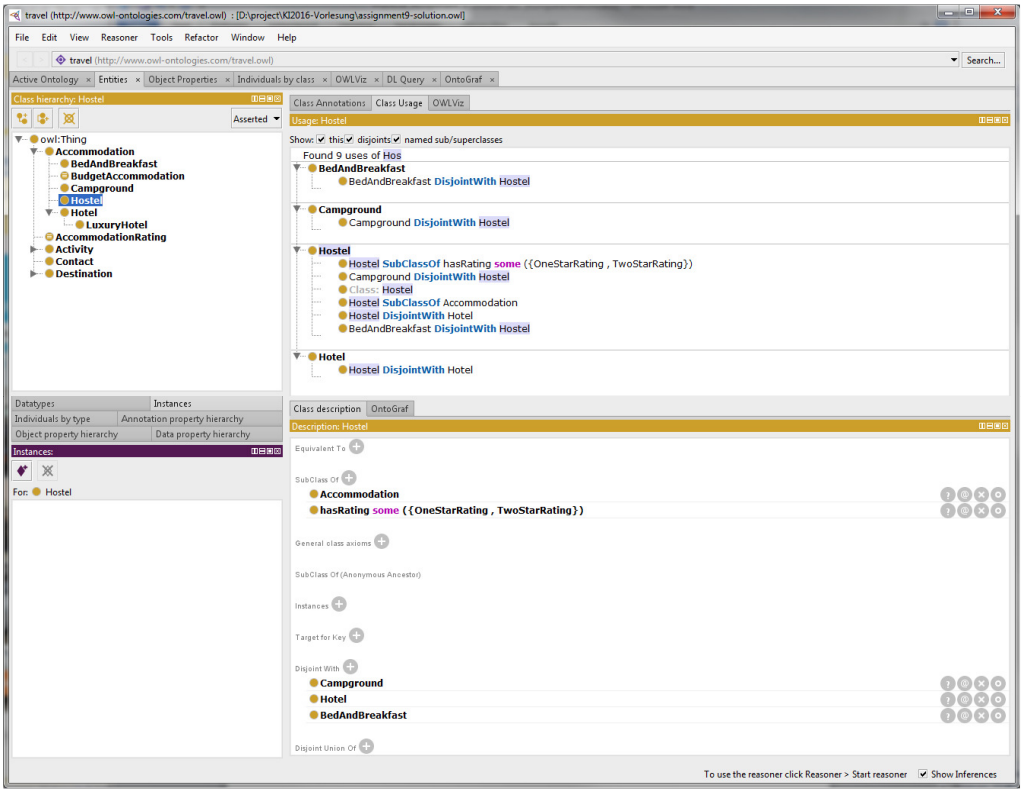
   hasAccommodation (Destination, Accommodation),
   has Activity (Destination, Activity),
   hasContact (Activity, Contact),
   hasPart (Destination, Destination),
   hasRating (Accommodation, AccommodationRating),
   isOfferedAt (Activity, Destination)

   b) *isOfferedAt* is the reverse relation to *hasActivity*. (0.5 points)
4) There are 4 data properties in the ontology: (1 point)

   hasCity (Contact, xsd:String)
   hasEMail (Contact, xsd:String)
   hasStreet (Contact, xsd:String)
   hasZipCode (Contact, xsd:int)

5) *Capital* is inferred as subclass of *RetireeDestination*, *NationalPark* is inferred as subclass of *BackpackersDestination* (1 point)

6) (2 points, 1 for the correct subclass, 1 for the correct disjoints)



Exercise 3: [Production Systems] (6 Points)
The following production system is designed to calculate the first 6 square numbers.
Explanations:

- MULTIPLY(X, Y) calculates the product of X and Y,
- SUCC(X) calculates the successor of X (in other words: X+1),
- PRINT(X) prints out the value of X on the console,
- END, RES and COUNT are variables, the notion (P X) means that variable P has the value X, where X is a number in WM entries or a label in rules,
- START, CALC, PRINTOUT, and SHIFT are static labels which may be added to (or deleted from) the working memory by the ADD (or DELETE) operator.
- REPLACE(X, Y, Z) means: replace all occurrences of X in term Z by Y, e.g., REPLACE(A, B, (COUNT A)) changes (COUNT A) into (COUNT B)

```
WM: (START) (COUNT 0) (END 6)

R1: (START) & (COUNT X1) →
     DELETE (START), ADD (RES 0), ADD (SHIFT),
R2: (END X1) & (COUNT X1) → STOP
R3: (SHIFT) & (COUNT X1) & (RES X2) →
     DELETE (SHIFT), ADD (CALC),
     REPLACE (X1, SUCC(X1),(COUNT X1)),
R4: (CALC) & (COUNT X1) & (RES X2) →
     DELETE (CALC), ADD (PRINTOUT),
     REPLACE (X2, MULTIPLY (X1, X1), (RES X2)),
R5: (PRINTOUT) & (COUNT X1) & (RES X2) →
     DELETE (PRINTOUT), PRINT (X2)
```

1) Calculate the output of this production system until no more rules are left to fire. For each cycle, write down the contents of the WM, the conflict set, the concrete instantiation of the firing rule, the output (if there is any) and the WM resulting after rule application. (3 points)

Example:
WM: (START) (COUNT 0) (END 6)
Conflict set: {R1}
Instantiation: [R1 ((START) (COUNT 0))]
Output:
WM: (RES 0) (SHIFT) (COUNT 0) (END 6)
Conflict set: …

2) Obviously there is an error in the rules above: Rule R2 is never going to fire. Change rule R5 in such a way that the cycle R3, R4, R5 is repeated until R2 fires. (2 points)

3) The rules above are independent of the chosen conflict resolution strategy. Why? (1 point)

Sample Solution:

1.

Step1:
WM: (START) (COUNT 0) (END 6)
Conflict Set: {R1}
Instantiation: [R1 ((START) (COUNT 0))]
Output:

Step2:
WM: (RES 0) (SHIFT) (COUNT 0) (END 6)
Conflict Set: {R3}
Instantiation: [R3 ((SHIFT) (COUNT 0) (RES 0))]
Output:

Step3:
WM: (RES 0) (CALC) (COUNT 1) (END 6)
Conflict Set: {R4}
Instantiation: [R4 ((CALC) (COUNT 1) (RES 0))]
Output:

Step4:
WM: (RES 1) (PRINTOUT) (COUNT 1) (END 6)
Conflict Set: {R5}
Instantiation: [R5 ((PRINTOUT) (COUNT 1) (RES 1))]
Output: 1

Step5:
WM: (RES 1) (COUNT 1) (END 6)
Conflict Set: { }
Instantiation: []
Output:

2. The Rule has to be changed in such a way that after printing the result a new shift step begins:

R5: (PRINTOUT) & (COUNT X1) & (RES X2) →
DELETE (PRINTOUT), PRINT (X2)
ADD (SHIFT)

3. The rules are independent of the chosen conflict resolution strategy because every rule (or step in the calculation) has a special "label" (e.g. CALC, START, etc.). These labels are always deleted after a rule has fired and a new label is added to trigger firing the next rule in the calculation process.

Submission instruction: Please submit your solution in paper on Wednesday, 13. July 2016, 16:00-16:15 in the lecture room.