Universität Freiburg
Institut für Informatik
Michael Meier
Fang Wei

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg
Tel. (0761) 203-8126
Tel. (0761) 203-8125

Webbasierte Informationssysteme
Winter Term 2010/2011
November 24, 2010

Fifth Exercise Set: SPARQL and OWL

Exercise 1

Consider the RDF database

D := { (_:P1,rdf:type,Person), (_:P1,name,“Pete”), (_:P1,age,“17”), (_:P1,email,“pete@abc.com”),
(_:P2,rdf:type,Person), (_:P2,name,“John”), (_:P2,email,“john@abc.com”),
(_:P3,rdf:type,Person), (_:P3,name,“Sue”), (_:P3,age,“21”),
(_:P1,knows,_:P2), (_:P1,knows,_P3), (_:P2,knows,_:P1), (_:P2,knows,_:P3) }.

Draw the RDF graph. Evaluate the following SPARQL queries and phrase their semantics in words. Assume that every of the following queries is preceeded by the necessary namespace definitions and the SELECT * solution format.

a) `{ ?p rdf:type Person. ?p age ?age. FILTER (?age>20) }`

b) `{ { ?p rdf:type Person. ?p name ?name. } OPTIONAL { ?p age ?age . } }`

c) `{ { ?p rdf:type Person. ?p age ?age. } UNION { ?p rdf:type Person. ?p email ?email. } }`

d) `{ { ?p rdf:type Person. OPTIONAL ?p email ?email. } FILTER (!bound(?email)) }`

Verify your results using the ARQ SPARQL engine. A short installation instruction, the above RDF document D, and example query a) are provided at the exercise page of the lecture homepage.

Exercise 2

Consider the RDF database D from the previous exercise. Specify the following requests as SPARQL queries and indicate the final results obtained when evaluating them on document D.

a) All pairs of distinct persons that have a common friend (i.e., it must hold that the intersection of persons they know is non-empty).

b) The names of all persons that know at least one person or are younger than 20 years. If present, the email address and, also if present, the age of this person should be included in the result.

c) Construct a new graph using the `CONSTRUCT` form that contains all persons (including their names) that know at least two persons.

d) Write a SPARQL `ASK` query that – given any document D as input – returns *yes* if and only if the constraint

$$\forall p(D(p, \mathit{rdf{:}type}, \mathit{Person}) \rightarrow \exists n\ D(p, \mathit{name}, n))$$

is violated (we interprete the RDF database D here as a ternary relation $D(\mathit{subject}, \mathit{predicate}, \mathit{object})$ that contains all RDF triples).

e) Write a SPARQL `ASK` query that – given any document D as input – returns *yes* if and only if the constraint

$$\forall p_1, p_2, n\ (D(p_1, \mathit{rdf{:}type}, \mathit{Person}), D(p_2, \mathit{rdf{:}type}, \mathit{Person}), D(p_1, \mathit{name}, n), D(p_2, \mathit{name}, n) \rightarrow p_1 = p_2)$$

is violated (we interprete the RDF database D here as a ternary relation $D(\mathit{subject}, \mathit{predicate}, \mathit{object})$ that contains all RDF triples).

Verify your results using the ARQ engine.

Exercise 3
The concept VegetarischePizza is to be defined. Which of the following definitions is suited best?

(a) $\text{VegetarischePizza} \equiv \text{Pizza} \sqcap \neg\exists\, \text{hatZutat}.(\text{Fleisch} \sqcap \text{Fisch})$

(b) $\text{VegetarischePizza} \equiv \text{Pizza} \sqcap \forall\, \text{hatBelag}.(\neg\text{Fleisch} \sqcup \neg\text{Fisch})$

(c) $\text{VegetarischePizza} \equiv \text{Pizza} \sqcap \neg\exists\, \text{hatBelag}.\text{Fleisch} \sqcap \neg\exists\, \text{hatBelag}.\text{Fisch}$

(d) $\text{VegetarischePizza} \equiv \text{Pizza} \sqcap \exists\, \text{hatBelag}.\neg\text{Fleisch} \sqcap \exists\, \text{hatBelag}.\neg\text{Fisch}$

(e) $\text{VegetarischePizza} \equiv \text{Pizza} \sqcap \forall\, \text{hatZutat}.(\neg\text{Fleisch} \sqcap \neg\text{Fisch})$

Exercise 4
Let the following ontology be given:

$\text{hatBelag} \sqsubseteq \text{hatZutat}$
$\text{Gemüse} \sqcap \text{Käse} \sqsubseteq \bot$
$\text{Gemüse} \sqcap \text{Fleisch} \sqsubseteq \bot$
$\text{Gemüse} \sqcap \text{Fisch} \sqsubseteq \bot$
$\exists\text{hatBelag}.\top \sqsubseteq \text{Pizza}$
$\text{Käse} \sqcap \text{Fleisch} \sqsubseteq \bot$
$\text{Käse} \sqcap \text{Fisch} \sqsubseteq \bot$
$\text{Fleisch} \sqcap \text{Fisch} \sqsubseteq \bot$
$\top \sqsubseteq \forall\text{hatBelag}.\text{PizzaBelag}$

Consider the additional class definitions:

$$\begin{aligned}
\text{KäsePizza} &\equiv \text{Pizza} \sqcap \exists\, \text{hatBelag}.\text{Käse}\\
\text{PizzaSpinat} &\equiv \exists\, \text{hatBelag}.\text{Spinat} \sqcap \exists\, \text{hatBelag}.\text{Käse} \sqcap\\
&\quad\ \forall\, \text{hatBelag}.(\text{Spinat} \sqcup \text{Käse})\\
\text{PizzaCarnivorus} &\equiv \text{Pizza} \sqcap \forall\, \text{hatBelag}.(\text{Fleisch} \sqcap \text{Fisch})\\
\text{LeerePizza} &\equiv \text{Pizza} \sqcap \neg\exists\, \text{hatBelag}.\top
\end{aligned}$$

- Which of these pizza classes are recognized as a subclass of VegetarischePizza by an OWL reasoner? (With respect to a correct definition of VegetarischePizza from the previous exercise.)
- The classification from bullet one shows that some classes of pizzas do not model the desired real world concept. How could the definitions be corrected?
- How would the result in bullet one change if we would use $\sqsubseteq$ instead of $\equiv$ in the definition of VegetarischePizza?

Due by: December 1, 2010 before the tutorial starts.