UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580/INF4580 — Semantic Technologies
Day of examination:	12 June 2013
Examination hours:	14:30 – 18:30

This problem set consists of 10 pages.

Appendices:	None
Permitted aids:	Any printed or written course material

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

The exam consists of five questions with equal weight.

Problem 1 RDF (20 %)

(a) Consider the "almost-Turtle" document below:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix pol: <http://eksempel.politikk.no/> .

"Jens Stoltenberg" a pol:Politician ;
       foaf:knows [a pol:Politician ;
                   foaf:name "Siv Jensen"] ;
       _:precededBy pol:KjellMagne ;
       "born" "16 March 1959"^^pol:date .
```

Find 4 mistakes, i.e. give 4 reasons why this is *not* a valid Turtle representation of an RDF graph.

(b) Given the `pol:` namespace prefix from above, and the vocabulary

pol:member for membership in a political party

pol:name for the name of a person

pol:Citizen for the class of all citizens,

pol:Party for the class of all parties,

(Continued on page 2.)

pol:Politician for the class of all politicians.

Express the following as a set of triples, written in Turtle syntax.

> There is a citizen who is member of a party that has a member who is a politician with name Jens Stoltenberg. The same citizen is also member of a party that has a member who is a politician with name Siv Jensen.

NOTE: use blank nodes instead of inventing URIs, whenever the question does not give you the URI of a resource.

(c) Give a graphical representation of the triples from (b)

Problem 2 SPARQL (20 %)

Consider an RDF document that contains information about projects: the tasks to be performed, the people who use time on the tasks, and the companies employing people who perform those tasks.

The data uses the following classes:

prj:Task a task that needs to be performed

prj:Person a person who can perform a task

prj:Company a company who employs persons

prj:Performance a resource describing how many hours someone used on a given task.

and the following properties:

rdfs:label links companies, persons, and tasks to their names

prj:employs links companies to their employees

prj:competence links companies to the tasks they are competent to perform.

prj:performs links a person to a "performance."

prj:perfTask links a "performance" to the task performed

prj:perfHours links a "performance" to the number of hours use on the task (an `xsd:decimal`)

(Continued on page 3.)

For instance, to say that person `prj:bob` uses 4 hours on task `prj:building`, we can use the triples

```
  prj:bob prj:performs [a prj:Performance ;
                        prj:perfTask prj:building ;
                        prj:perfHours "4"^^xsd:decimal ] .
```

Here is some example data for parts of a house building project:

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix prj:    <http://example.org/project#> .

prj:sewer a prj:Task ; rdfs:label "Install sewerage system" .
prj:gas   a prj:Task ; rdfs:label "Install gas pipes" .
prj:elect a prj:Task ; rdfs:label "Install electricity" .

prj:leaky  a prj:Company ;
           rdfs:label "Leaky & Sons Plumbers Ltd" ;
           prj:competence prj:sewer, prj:gas ;
           prj:employs prj:lenny, prj:louise .

prj:fizzle a prj:Company ;
           rdfs:label "Fizzle Wiring Company" ;
           prj:competence prj:elect ;
           prj:employs prj:fiona, prj:fritz .

prj:lenny a prj:Person ; rdfs:label "Lenny" .
prj:louise a prj:Person ; rdfs:label "Louise" .
prj:fiona a prj:Person ; rdfs:label "Fiona" .
prj:fritz a prj:Person ; rdfs:label "Fritz" .

prj:fritz prj:performs [a prj:Performance ;
                        prj:perfTask prj:sewer ;
                        prj:perfHours "1"^^xsd:decimal ] .

prj:louise prj:performs [a prj:Performance ;
                        prj:perfTask prj:gas ;
                        prj:perfHours "2.5"^^xsd:decimal ] .
...
```

Imagine that we have data about many more tasks, companies, employees,

(Continued on page 4.)

and performances. In the queries you write to answer the following questions, you are not required to write out the PREFIX declarations.

(a) Write a query that lists the names and used hours for all tasks that have been performed by employees of the company named "Rudolf's Roofs."

(b) Write a query that lists the names of tasks that have been performed by employees of two (or more) different companies, along with the names of both employees.

(c) Write a query that lists the names of companies where some employee has performed a task that took more than 10 hours.

(d) Write a query that lists the names of company, employee, and task for all cases where an employee of a company has performed a task that the company does not have competence for.

(e) Write a query that lists the name of each company together with the total number of hours spent on tasks by its employees.

Problem 3 RDFS Reasoning (20 %)

Idea: Result of SPARQL query under RDFS reasoning? Explain forward reasoning, backward reasoning.

Let the following set of triples be given:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix v: <http://example.org/vocab/> .
@prefix h: <http://example.org/myhouse/> .
```

(1) `v:Door rdfs:subClassOf v:Component .`

(2) `v:Wall rdfs:subClassOf v:Component .`

(3) `v:hasPart rdfs:domain v:Component .`

(4) `v:hasPart rdfs:range v:Component .`

(5) `v:hasDoor rdfs:subPropertyOf v:hasPart .`

(6) `v:hasDoor rdfs:range v:Door .`

(7) `h:house v:hasPart h:frontWall .`

(8) `h:house v:hasPart h:backWall .`

(Continued on page 5.)

(9) `h:frontWall v:hasDoor h:frontDoor .`

(10) `h:backWall v:hasPart h:backDoor .`

(11) `h:backDoor rdf:type v:Door .`

(a) With RDFS reasoning ("under an RDFS reasoning regime"), what are the answers to the following SPARQL query? (Prefixes as before)

```
SELECT ?c ?d WHERE {
  ?c a v:Component ;
     v:hasPart ?d .
  ?d a v:Door .
}
```

Describe the forward reasoning needed on the triples to produce these answers.

(b) Explain the idea of "backward chaining."

(c) Consider the triple

```
v:hasDoor rdfs:domain v:Component .
```

and answer the following two questions: Is it entailed by the triples? Is it derivable using the RDFS rules? Please explain your answers.

Problem 4 Description logics/OWL (20 %)

Express each of the following statements as one or more OWL axioms. You may use the following class and property (role) names without namespaces:

- Classes: CelesitalBody, Sun, Moon, Planet
- Properties: orbits, mass

"orbits" is an object property that specifies which objects more around which others. For instance, `:europa :orbits :jupiter` means that the Europa (one of Jupiter's moons) orbits Jupiter.

"mass" is a datatype property giving an object's mass in multiples of the Earth's mass. For instance, `:pluto :mass "0.002"^^xsd:double` means that Pluto has 0.002 times the mass of the Earth.

You may use DL syntax, Manchester syntax, or any other OWL syntax.

(Continued on page 6.)

(a) Suns, moons, and planets are celestial bodies, and any celestial body can be at most one of these three types.

(b) Every celestial body orbits at most one other celestial body.

(c) A sun does not orbit any celestial body.

(d) A moon is a celestial object that orbits an object that is not a sun.

(e) A planet is defined to be an object that orbits a sun and has a mass of at least 0.01 times the mass of the earth.

Problem 5 RDF and OWL semantics (20 %)

(a) Give an interpretation with a non-empty $child^{\mathcal{I}}$ that satisfies the axiom

$$Mum \sqsubseteq \exists child.Boy \sqcap \forall child.\neg Boy$$

(b) Give a short proof sketch of the fact that the following two axioms are equivalent:

$$A \sqcup B \sqsubseteq \neg C$$

$$C \sqsubseteq (\neg A) \sqcap (\neg B)$$

I.e. (1) an interpretation that makes the first axiom true makes the second axiom true, and (2) an interpretation that makes the second axiom true makes the first axiom true.

(c) Give an interpretation $\mathcal{I}$ that satisfies the triple `_:x foaf:knows _:x`, but not the OWL axiom saying that `foaf:knows` is reflexive.

(Continued on page 7.)