UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580 — Semantic Technologies
Day of examination:	10. juni 2010
Examination hours:	09:00 – 12:00

This problem set consists of 11 pages.

Appendices:	None
Permitted aids:	Any printed course material

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

Problem 1 RDF (10 %)

Consider the RDF document below:

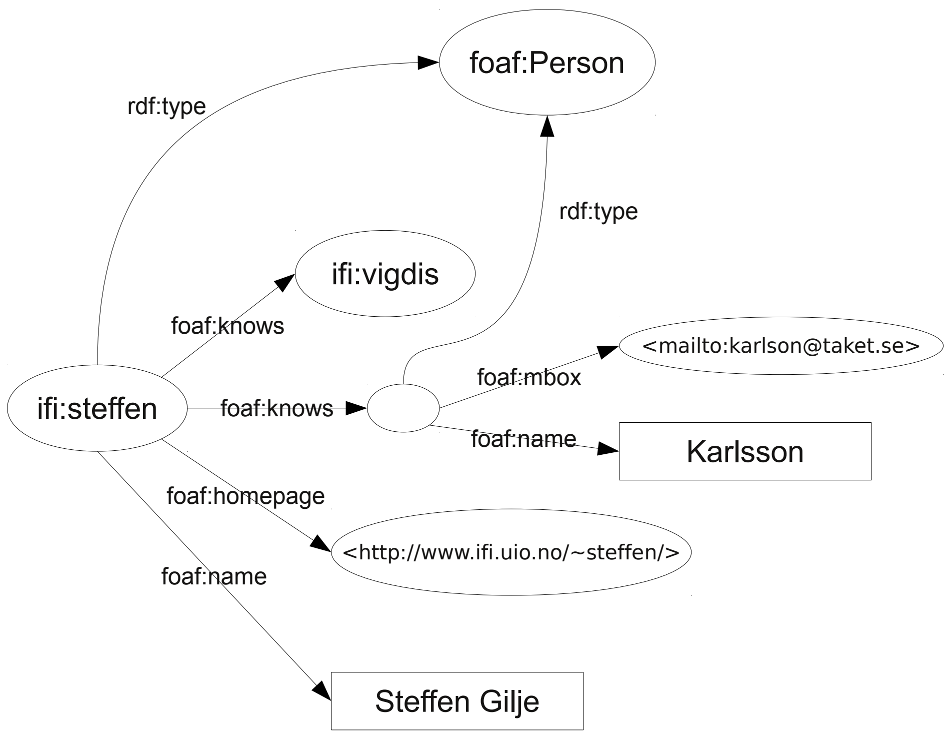
```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ifi: <http://www.ifi.uio.no/people#> .


ifi:steffen a foaf:Person;
    foaf:knows ifi:vigdis,
               [a foaf:Person;
                  foaf:mbox <mailto:karlsson@taket.se>;
                  foaf:name "Karlsson" ] ;
    foaf:name "Steffen Gilje";
    foaf:homepage <http://www.ifi.uio.no/~steffen/>.
```

Answer the following questions:

(a) Draw a graph representation of this RDF document.

Answer:

(Continued on page 2.)



(b) Which URI is used to identify Steffen Gilje in this document?

Answer:

```
http://www.ifi.uio.no/people#steffen
```

(c) What kind of resource represents Karlsson in this document?

Answer:

A blank node.

(d) The `foaf:interest` property is often used to represent one or more of the interests of a `foaf:Person`. One usually does so by pointing to a resource that indicates a `foaf:Document` whose `foaf:topic` can be said to characterise that interest.

Add statements to the document above that say the following:

1. that one `foaf:interest` of Steffen is `http://workingontologist.com`,
2. that the resource referred to by `http://www.workingontologist.com` is a `foaf:Document`, and that its `foaf:topic` is `"Semantic Web"`.

(Continued on page 3.)

Answer:

```
ifi:steffen foaf:interest <http://workingontologist.com> .
<http://workingontologist.com> a foaf:Document ;
                               foaf:topic "Semantic Web" .
```

Problem 2 SPARQL (20 %)

Consider the following RDF document that contains information about NASA space probe missions.

```
@prefix nasa: <http://www.nasa.gov/vocab#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

nasa:voyager a nasa:SpaceProbe;
             nasa:objective nasa:jupiter, nasa:saturn,
                            nasa:uranus, nasa:neptune.

nasa:mariner9 a nasa:SpaceProbe;
              nasa:objective nasa:mars.

nasa:spirit a nasa:SpaceProbe;
            nasa:objective nasa:mars.

nasa:pioneer10 a nasa:SpaceProbe;
               nasa:objective nasa:jupiter.

nasa:jupiter a nasa:Planet;
             nasa:moon nasa:io, nasa:ganymede, nasa:callisto;
             nasa:distanceFromSun "778"^^xsd:double .

nasa:saturn  a nasa:Planet;
             nasa:distanceFromSun "1426"^^xsd:double .

nasa:neptune a nasa:Planet;
             nasa:moon nasa:triton, nasa:proteus, nasa:naiad;
             nasa:distanceFromSun "4503"^^xsd:double .

nasa:uranus a nasa:Planet;
            nasa:distanceFromSun "2876"^^xsd:double .

nasa:mars a nasa:Planet;
          nasa:distanceFromSun "227"^^xsd:double .
```

(Continued on page 4.)

Distance from the sun is measured in millions of kilometers. Write a SPARQL query that retrieves

(a) all spaceprobes,

Answer:

```
SELECT ?probe WHERE {
  ?probe a nasa:SpaceProbe .
}
```

(b) all moons,

Answer:

```
SELECT ?moon WHERE {
  ?planet nasa:moon ?moon .
}
```

(c) for each planet that has one or more moons; the planet and its moons

Answer:

```
SELECT ?planet ?moon WHERE {
  ?planet a nasa:Planet ;
          nasa:moon ?moon .
}
```

Minus 10% for forgetting to say that `?planet` must be a `nasa:Planet` here and in the following.

(d) a planet and its distance from the sun,

Answer:

```
SELECT ?planet ?dist WHERE {
  ?planet a nasa:Planet ;
          nasa:distanceFromSun ?dist .
}
```

Adding `LIMIT 1` to reflect ‘a’ is OK, but not needed for 100%.

(e) all planets that lie further than 2000 million kilometers from the sun,

Answer:

(Continued on page 5.)

```
SELECT ?planet WHERE {
  ?planet a nasa:Planet ;
          nasa:distanceFromSun ?dist .
  FILTER(?dist > 2000)
}
```

(f) all planets that have been visited by (i.e. been the objective of) more than one space probe

Answer:

```
SELECT DISTINCT ?planet WHERE {
  ?planet a nasa:Planet .
  ?pr1 a nasa:SpaceProbe ; nasa:objective ?planet .
  ?pr2 a nasa:SpaceProbe ; nasa:objective ?planet .
  FILTER (?pr1 != ?pr2)
}
```

DISTINCT not required for 100%.

Problem 3 Manipulating RDF programmatically (15 %)

Imagine that you are writing a web application to assist people in creating a FOAF file for themselves (admittedly there are many such already). The web page is supposed to present the user with a set of input fields, e.g.

Title (Mr, Mrs, Dr, etc)	▷
First Name	▷
Last Name	▷
Nickname	▷
Your Email Address	▷
Homepage	▷
Your Picture	▷
Phone Number	▷

and the application is supposed to constructs a FOAF graph from the input it receives through these fields.

The code below sketches one way one may go about implementing the back-end of this application, that is, of constructing the FOAF graph from the input fields. Notice the consecutively numbered points in the code marked with asterisks. These are for you to fill in, in accordance with the instructions below:

(Continued on page 6.)

```
final private String foafNS = "http://xmlns.com/foaf/0.1/";
private Model foafModel;
private Resource thePerson;

public void initFoafModel(String foafFileURL){
    foafModel = * Point 1 *
    thePerson = foafModel.createResource(foafFileURL + "#me");
    Resource personType = foafModel.createResource(foafNS + "Person");

    *Point 2*
}

public void addTitle(String title){

    * Point 3 *
}

public void addMail(String mailAdress){

    * Point 4 *
}
```

Instructions:

(a) Construct an empty `Model` object at point 1 using Jena's `ModelFactory` class.

Answer:

```
ModelFactory.createDefaultModel()
```

(b) At point 2, add a statement to the graph held by `foafGraph` that says that the resource held by `thePerson` is one whose `rdf:type` is `foaf:Person`.

Answer:

```
String rdfNS = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
Property rdfType = foafModel.createProperty(rdfNS+"type");
thePerson.addProperty(rdfType, personType);
```

(c) Complete the method at point 3 to associate a `foaf:title` with `thePerson`.

Answer:

```
Property foafTitle = foafModel.createProperty(foafNS+"title");
thePerson.addProperty(foafTitle, title);
```

(Continued on page 7.)

(d) Complete the method at point 4 to associate a `foaf:mbox` with `thePerson`.

Answer:

```
Property foafMbox = foafModel.createProperty(foafNS+"mbox");
Resource mbox = foafModel.createResource("mailto:"+mailAdress);
thePerson.addProperty(foafMbox, mbox);
```

Problem 4 RDFS Reasoning (20 %)

Below is an excerpt from an RDFS document that keeps track of the cars and deliveries of a delivery service. We assume that the namespaces `dlv`, `rdf` and `rdfs` are given.

```
dlv:Car rdf:type rdfs:Class .
dlv:AssignedCar rdf:type rdfs:Class .
dlv:AssignedCar rdfs:subsetOf dlv:Car .
dlv:destination rdfs:domain dlv:AssignedCar .
dlv:destination rdfs:range dlv:City .
dlv:removalist rdfs:range dlv:Employee .
dlv:executive rdfs:range dlv:Employee .
dlv:mover rdfs:subPropertyOf dlv:removalist .
dlv:driver rdfs:subPropertyOf dlv:removalist .

dlv:AV43634 dlv:destination dlv:Trondhjem;
            dlv:driver dlv:BjarneBerg;
            dlv:mover  dlv:MagneMo;
            dlv:mover  dlv:HangHiu;
            dlv:executive dlv:LinnLarsson .

dlv:HJ14522 dlv:destination dlv:Grorud;
            dlv:driver dlv:LinnLarsson;
            dlv:mover  dlv:HelgeHareide;
            dlv:executive dlv:LinnLarsson .

dlv:SU56782 rdf:type dlv:Car .
```

For each of the triples below, determine whether it is entailed by the statements in the document. Justify your answer.

(a) `dlv:AV43634 rdf:type dlv:AssignedCar .`

Answer:

(Continued on page 8.)

Yes, since the `rdfs:destination` has `rdfs:domain` `dlv:AssignedCar`, and `dlv:AV43634` has a `dlv:destination`

(b) `dlv:AV43634 rdf:type dlv:Car .`

Answer:

Yes, because of (a) and because `dlv:AssignedCar` is a `rdfs:subClassOf` `dlv:Car`.

(`rdfs:subsetOf` in the problem is a typo. Pointing this out and saying the triple can't be derived is also OK)

(c) `dlv:SU56782 rdf:type dlv:AssignedCar .`

Answer:

No, it is only known that it is a `dlv:Car`

(d) `dlv:Trondhjem rdf:type dlv:City .`

Answer:

Yes, because `dlv:AV43634 dlv:destination dlv:Trondhjem` and `dlv:destination rdfs:range dlv:City` .

(e) `dlv:Grorud rdf:type dlv:City .`

Answer:

Yes, for the same reason as before, for `dlv:HJ14522`

(f) `dlv:HJ14522 dlv:removalist dlv:LinnLarsson .`

Answer:

Yes, since `dlv:LinnLarsson` is `dlv:driver` for `dlv:HJ14522`, and `dlv:driver` is a `rdfs:subPropertyOf` `dlv:removalist`.

(g) `dlv:AV43634 dlv:removalist dlv:LinnLarsson .`

Answer:

No, `dlv:LinnLarsson` is `dlv:executive` for `dlv:AV43634` but that is not a sub-property of `dlv:removalist`.

(h) `_:blank dlv:removalist dlv:MagneMo .`

Answer:

Yes. For this to be true, one needs to be able to assign a resource in the given graph to the blank node, such that the resulting triple occurs in the graph. `dlv:AV43634` does the trick.

(i) `_:blank dlv:removalist _:blank .`

Answer:

No. This is only one blank node with an edge to itself. The triples contain no `dlv:removalist` edge from any node to itself, and none can be derived.

(Continued on page 9.)

(j) `dlv:destination rdf:type rdf:Property .`

Answer:

Yes. `dlv:destination` occurs as the predicate in some triples, so it has `rdf:type rdf:Property` according to RDFS semantics.

Problem 5 Description logics/OWL (20 %)

Express the following sentences in DL-notation, using the the class names `Vehicle`, `Car`, `ElectricCar`, `HybridCar`, `Engine`, `Wheel`, `CombustionEngine`, `ElectricEngine` and the role names `hasPart` and `hasEngine`.

(a) A car is a vehicle with an engine.

Answer:

$Car \sqsubseteq Vehicle \sqcap \exists hasEngine.Engine$

$\exists hasEngine.\top$ is also OK. Forgetting $Vehicle$ here and in the other questions gives minus 10%. Using $\equiv$ instead of $\sqsubseteq$ is OK.

(a) A car is a vehicle with a combustion engine or an electric engine.

Answer:

$Car \sqsubseteq Vehicle \sqcap \exists hasEngine.(CombustionEngine \sqcup ElectricEngine)$

(b) Every car has exactly four wheels as part.

Answer:

$Car \sqsubseteq =_4 hasPart.Wheel$

(c) A four-wheeled vehicle with a combustion engine or an electric engine is a car.

Answer:

$Vehicle \sqcap =_4 hasPart.Wheel \sqcap$
$\exists hasEngine.(CombustionEngine \sqcup ElectricEngine)$
$\sqsubseteq Car$

(d) An electric car is a car that has only an electric engine

Answer:

$ElectricCar \sqsubseteq Car \sqcap \forall hasEngine.ElectricEngine$

Attempts at saying that there is at least one or exactly one engine are OK, but not required.

(Continued on page 10.)

(e) A hybrid car is a car that has a combustion engine as well as an electric engine.

Answer:

$HybridCar \sqsubseteq Car \sqcap (\exists hasEngine.CombustionEngine) \sqcap (\exists hasEngine.ElectricEngine)$

Problem 6 Description logic/OWL semantics (15 %)

Consider the ontology consisting of the following axioms:

Axiom 1: $PetShop \sqsubseteq Shop \sqcap \exists offers.Animal$

Axiom 2: $Vertebrate \sqsubseteq Animal$

Axiom 3: $Invertebrate \sqsubseteq Animal$

Axiom 4: $Vertebrate \sqsubseteq \neg Invertebrate$

(a) Is it true, according to this ontology, that an animal can be a vertebrate *as well as* an invertebrate? Justify your answer with reference to the axioms.

Answer:

No. The disjointness axiom 4 says that a vertebrate cannot be an invertabrate.

(a) The ontology does not entail that the things on offer in a pet shop are either vertebrates or invertebrates. To show this, complete the countermodel below by assigning an appropriate subset of $\Delta^{\mathcal{I}}$ to $Animal^{\mathcal{I}}$ and an appropriate relation over $\Delta^{\mathcal{I}}$ to $offers^{\mathcal{I}}$

$\Delta^{\mathcal{I}}$	$= \{petsRus, fido, jellybean\}$
$Vertebrate^{\mathcal{I}}$	$= \{fido\}$
$Invertebrate^{\mathcal{I}}$	$= \{jellybean\}$
$PetShop^{\mathcal{I}} = Shop^{\mathcal{I}}$	$= \{petsRus\}$
$offers^{\mathcal{I}}$	$=$
$Animal^{\mathcal{I}}$	$=$

Answer:

To build a countermodel, we need a pet shop (and there is only *petsRus* in the domain) to sell something that is neither a vertebrate or an invertebrate. *fido* and *jellybean* are fixed to be respectively vertebrate and invertebrate. The only element left in the domain is

(Continued on page 11.)

$petsRus$ itself. So we can define $offers^{\mathcal{I}} = \{(petsRus, petsRus)\}$ and since a pet shop needs to offer at least one animal, $Animal^{\mathcal{I}} = \{petsRus, fido, jellybean\}$.

Alternatively, $petsRus$ could *also* offer one or both of $fido$ and $jellybean$, e.g. $offers^{\mathcal{I}} = \{(petsRus, petsRus), (persRus, fido)\}$, in which case $petsRus$ does not have to be (but can) be an element of $Animal^{\mathcal{I}}$.

Answers giving something like a model (set of pairs for $offers^{\mathcal{I}}$, etc.) but wrong, get 20%.

Answers giving a counter-model but to a misunderstood statement, or by changing the given domain get 80%.

(c) Remedy this defect (which it may reasonably be taken to be) by adding a covering axiom that applies to animals.

Answer:

$Animal \sqsubseteq Vertebrate \sqcup Invertebrate$