UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580 — Semantic Technologies
Day of examination:	14 June 2011
Examination hours:	09:00 – 13:00

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 %)

Consider the RDF document below:

```
@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 foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dbp: <http://dbpedia.org/resource/> .
@prefix dbp-owl: <http://dbpedia.org/ontology/> .

dbp:Ivo_Caprino a foaf:Person;
    dbp-owl:birthPlace _:x ;
    dbp-owl:deathPlace _:x ;
    dbp-owl:birthDate "1920-02-17"^^xsd:date ;
    dbp-owl:deathDate "2001-02-08"^^xsd:date .

_:x a dbp-owl:Place ;
      rdfs:label "Oslo"@no, "Oslo"@en ;
      dbp-owl:country dbp:Norway .
```

Answer the following questions:

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

(Continued on page 2.)

(b) What kind of RDF node does `_:x` denote in this document?

(c) What kind of RDF node does `"1920-02-17"^^xsd:date` denote in this document?

(d) The document mentions one resource of type `foaf:Person`. What does the document say about the name of this person?

(e) Add statements using the given `dbp-owl` properties and the FOAF vocabulary to say that there is a *person* who has the *name* "Fridtjof Nansen", and who is *born at* the same place as the person already mentioned in the document.

Answer:

(a) ...

(b) A blank node. (Representing something named "Oslo" in at least two languages and located in Norway.)

(c) A typed literal. Representing 17 February 1920.

(d) Nothing.

(e)
```
[a foaf:Person ;
 foaf:name "Fridtjof Nansen" ;
 dbp-owl:birthPlace _:x] .
```

Problem 2 SPARQL (20 %)

Consider an RDF document that contains information about bands, albums they released, and tracks contained on these albums.

Band names are coded using `foaf:name`, whereas album and track titles are represented with `dc:title`. The albums released by a band are given by `foaf:made`, and the year an album was released with `dc:created`. Tracks on albums are given with `m:hasTrack`. Here is an example of some information about one band and some of its recordings:

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc:   <http://purl.org/dc/terms/> .
@prefix m:    <http://example.org/music#> .

m:stones a m:Band;
```

(Continued on page 3.)

```
    foaf:name "The Rolling Stones" ;
    foaf:made m:aftermath, m:exile, m:licks, m:bang .

m:aftermath a m:Album;
    foaf:name "Aftermath" ;
    dc:created "1966"^^xsd:integer ;
    m:hasTrack m:thumb, m:ladyJane .

m:exile a m:Album;
    foaf:name "Exile on Main St." ;
    dc:created "1972"^^xsd:integer .

m:licks a m:Album;
    dc:title "Forty Licks" ;
    dc:created "2005"^^xsd:integer ;
    m:hasTrack m:jackFlash, m:thumb, m:satisfaction .

m:thumb a m:Track; dc:title "Under My Thumb" .
m:ladyJane a m:Track; dc:title "Lady Jane" .
m:jackFlash a m:Track; dc:title "Jumping Jack Flash" .
m:satisfaction a m:Track; dc:title "Satisfaction" .
```

Imagine that we have data about many more bands, albums, and tracks. 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 of all albums by a band called "The Beatles".

(b) Write a query that lists the names of all bands who have released albums over a period of at least twenty years.

(c) Some albums ("compilations") contain tracks that have been published previously. Write a query to list the titles of all albums containing a track that is also contained in an album that was released earlier.

(d) Without using SPARQL 1.1 features or the BOUND function of SPARQL 1.0, it is not possible to write a query that lists all tracks that have *not* appeared on an earlier album. Say why such a query would be problematic from a semantic web perspective.

(e) If the same song was recorded by different bands, we consider that to be different tracks. In other words, if the same track (i.e. the same resource) appears on albums released by different artists, that's a mistake. To test whether our data contains any such mistakes, write a query to list the URI and name of all such tracks.

(Continued on page 4.)

(f) Bands not only make albums, they also make the tracks on the albums. Write a CONSTRUCT query that builds a triple *b* `foaf:made` *t* for each track *t* on an album released by the band *b*.

Answer: In all questions:

- The problem text says `dc:title` for album titles, but the example triples mistakenly use `foaf:name`. Answers using either are OK.

(a)
```
SELECT DISTINCT ?name WHERE {
  ?b a m:Band;
     foaf:name "The Beatles";
     foaf:made ?a;
  ?a a m:Album;
     dc:title ?name .
}
```

(b)
```
SELECT DISTINCT ?name WHERE {
  ?b a m:Band;
     foaf:name ?name;
     foaf:made ?a1, ?a2 .
  ?a1 a m:Album; dc:created ?y1 .
  ?a2 a m:Album; dc:created ?y2 .
  FILTER (?y1 - ?y2 >= 20)
}
```

(c)
```
SELECT DISTINCT ?title WHERE {
  [a m:Album;
   dc:title ?title;
   dc:created ?y1;
   m:hasTack ?t ] .
  [a m:Album;
   dc:created ?y2;
   m:hasTack ?t ] .
  FILTER (?y2 < ?y1)
}
```

(d) Due to the "open world assumption", there might be both albums not mentioned in the data, and tracks on albums which are not mentioned. Any number of triples will never entail that some track has *not* appeared on an earlier album, so any answer given by a query could be invalidated by additional information.

(e)
```
SELECT DISTINCT ?t ?name WHERE {
  ?t a m:Track; dc:title ?name .
```

(Continued on page 5.)

```
      ?b1 a m:Band; foaf:made [a m:Album; m:hasTrack ?t] .
      ?b2 a m:Band; foaf:made [a m:Album; m:hasTrack ?t] .
      FILTER (?b1 != ?b2)
    }
```

(f)
```
    CONSTRUCT {
      ?b foaf:made ?t .
    } WHERE {
      ?b a m:Band; foaf:made [a m:Album; m:hasTrack ?t] .
      ?t a m:Track .
    }
```

Problem 3 RDFS Reasoning (20 %)

Let the following set of triples be given:

```
@prefix r: <http://example.org/relationship/>
@prefix p: <http://example.org/people/>
```

(1) `r:relatedTo rdfs:domain r:Person .`
(2) `r:relatedTo rdfs:range r:Person .`
(3) `r:spouseOf rdfs:subPropertyOf r:relatedTo .`
(4) `r:spouseOf rdfs:subPropertyOf r:knows .`
(5) `r:wifeOf rdfs:subPropertyOf r:spouseOf .`
(6) `r:wifeOf rdfs:domain foaf:Woman .`
(7) `r:wifeOf rdfs:range foaf:Man .`
(8) `r:husbandOf rdfs:subPropertyOf r:spouseOf .`
(9) `r:husbandOf rdfs:domain r:Man .`
(10) `r:husbandOf rdfs:range r:Woman .`

(11) `p:bjorn r:husbandOf p:agnetha .`
(12) `p:agnetha r:knows p:anni-frid .`
(13) `p:anni-frid r:wifeOf p:benny .`
(14) `p:benny r:spouseOf p:anni-frid .`

For each of the following triples (or sets of triples, in (f)), either give a derivation using the rules of RDFS and simple entailment, or give a short explanation of why such a derivation does not exist. If no derivation exists,

(Continued on page 6.)

also indicate whether the statement is entailed or not (under the simplified RDF/RDFS semantics used in the course).

(a) `r:wifeOf rdfs:subPropertyOf r:relatedTo`

(b) `p:bjorn r:knows p:agnetha`

(c) `p:agnetha r:wifeOf p:bjorn`

(d) `p:bjorn rdf:type r:Person`

(e) `r:husbandOf rdfs:domain r:Person`

(f)
```
_:x r:spouseOf _:y .
_:y r:spouseOf _:x .
```

Answer:

(a) (a1) `r:wifeOf rdfs:subPropertyOf r:relatedTo` from (5) and (3) by rdfs5

(b) (b1) `p:bjorn r:spouseOf p:agnetha` from (11) and (8) by rdfs7

(b2) `p:bjorn r:knows p:agnetha` from (b1) and (4) by rdfs7

(c) This cannot be derived, since the axioms do not give a connection between `r:wifeOf` and `r:husbandOf`. In OWL, one could be declared to be the inverse of the other, but this is not possible in RDFS. The statement is not entailed by the given triples.

(d) (d1) `p:bjorn r:spouseOf p:agnetha` from (11) and (8) by rdfs7

(d2) `p:bjorn r:relatedTo p:agnetha` from (d1) and (3) by rdfs7

(d3) `p:bjorn rdf:type r:Person` from (d2) and (1) by rdfs2

(e) This cannot be derived, since there are no RDFS rules to derive new `rdfs:domain` triples. It is however entailed because if *x* `r:husbandOf` *y*, then *x* `r:relatedTo` *y*, and from (1), *x* `rdf:type r:Person`.

(f) (f1) `_:x r:wifeOf p:benny` from (14) by se2, `_:x`→`p:anni-frid`

(f2) `_:x r:wifeOf _:y` from (f1) by se1, `_:y`→`p:benny`

(f3) `_:x r:spouseOf _:y` from (f2) and (8) by rdfs7, `_:y`→`p:benny`

(f4) `_:y r:spouseOf p:anni-frid` from (14) by se2, reusing `_:y`→`p:benny`

(f5) `_:y r:spouseOf _:x` from (f4) by se1, reusing `_:x`→`p:anni-frid`
(f3) and (f5) are the required triples, so the derivation is successful.

(Continued on page 7.)

Problem 4 Description logics/OWL (20 %)

Consider information about movies and the involved actors and crew coded in the following way:

```
film:avatar a :Movie ;
      dc:title "Avatar"@en ;
      :director people:james ;
      :leading-actor people:sam ;
      :leading-actor people:zoe ;
      :supporting-actor people:michelle .

people:james a :Man ;
      foaf:name "James Cameron" .

people:sam a :Man ;
      foaf:name "Sam Worthington" .

people:zoe a :Woman ;
      foaf:name "Zoe Saldana" .

people:michelle a :Woman ;
      foaf:name "Michelle Rodriguez" .
```

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: Movie, Man, Woman, Person, LoveMovie, AnimalDocumentary, AnimatedMovie
- Properties: title, director, leading-actor, supporting-actor, actor, male-lead, female-lead

(a) Any man and any woman is a person, and any person is either a man or a woman, but nothing is both a man and a woman.

(b) Any "male lead" in a movie is a man.

(c) Any "male lead" or "female lead" is a leading actor, any leading actor or supporting actor is an actor. (Note: these are all roles linking movies to people.)

(d) A love movie is a movie with at least one male leading actor and one female leading actor, and all such movies are love movies.

(Continued on page 8.)

(e) A movie that has no actors is either an animal documentary or an animated movie.

Answer:

(a)

$$Person \equiv Man \sqcup Woman$$
$$Man \sqcap Woman \equiv \bot$$

Alternatively: explicit subset axioms for Man and Woman, other representations of disjointness.

(b) $Movie \sqsubseteq \forall male\text{-}lead.Man$

(c)

$$male\text{-}lead \sqsubseteq leading\text{-}actor$$
$$female\text{-}lead \sqsubseteq leading\text{-}actor$$
$$leading\text{-}actor \sqsubseteq actor$$
$$supporting\text{-}actor \sqsubseteq actor$$

Note that $\sqsubset$ denotes the sub-property relationship between these properties.

(d) $LoveMovie \equiv Movie \sqcap \exists leading\text{-}actor.Man \sqcap \exists leading\text{-}actor.Woman$
Alternatively: $LoveMovie \equiv Movie \sqcap \exists male\text{-}lead.\top \sqcap \exists female\text{-}lead.\top$

(e) $Movie \sqcap \neg\exists actor.\top \sqsubseteq AnimalDocumentary \sqcup AnimatedMovie$

Problem 5 RDF and OWL semantics (20 %)

(a) Remember that we defined the semantics of the triple

$$R \texttt{ rdfs:domain } C$$

to be such that it is valid in some DL-interpretation $\mathcal{I}$ if and only if

$$\text{dom } R^{\mathcal{I}} \subseteq C^{\mathcal{I}}$$

We saw that the same requirement can be expressed as an OWL axiom:

$$\exists R.\top \sqsubseteq C$$

Give an OWL axiom that additionally expresses that *all* elements of C are in the domain of R, i.e.

$$\text{dom } R^{\mathcal{I}} = C^{\mathcal{I}}$$

(Continued on page 9.)

(b) Every car has some part which is a motor. Given the classes `:Car` and `:Motor`, and the relation `:hasPart`, can this fact be expressed using the following triples?

```
:Car :hasPart _:x .
_:x a :Motor .
```

Briefly explain your answer. Give an OWL axiom that expresses that "every car has some part which is a motor."

(c) Given an individual `x`, a property `p`, and a class `C`, the triples

```
x   p _:b .
_:b a C   .
```

are equivalent to the following OWL axiom

$$(\exists \mathtt{p}.\mathtt{C})(\mathtt{x})$$

which asserts that `x` is a member of the concept $\exists$`p`.`C`. I.e. any DL-interpretation that satisfies the two triples satisfies the OWL axiom, and vice versa. Give a short proof sketch of this fact.

(d) Any motor of a car is a piece of machinery, as expressed by the OWL axiom

$$Car \sqsubseteq \forall motor.Machinery \tag{1}$$

This axiom does *not* entail that every car has a motor which is a piece of machinery, i.e.

$$Car \sqsubseteq \exists motor.Machinery \tag{2}$$

Give a counterexample $\mathcal{I}$ that satisfies (1), but not (2), by explicitly stating $\Delta^{\mathcal{I}}$, $Car^{\mathcal{I}}$, $Machinery^{\mathcal{I}}$, $motor^{\mathcal{I}}$.

Answer:

(a)

$$\exists R.\top \equiv C$$

Explanation:

$$(\exists R.\top)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{for some } y \in \top^{\mathcal{I}} = \Delta^{\mathcal{I}},\ \langle x, y\rangle \in R^{\mathcal{I}}\} = \mathsf{dom}\, R^{\mathcal{I}}$$

Therefore

$$\mathcal{I} \models \exists R.\top \equiv C \quad \text{iff} \quad \mathsf{dom}\, R^{\mathcal{I}} = C^{\mathcal{I}} \quad .$$

(Continued on page 10.)

(b) The given triples say that the `:Car` class itself has a motor, and not that every element of the class has one. In fact this is an example of triples not allowed under our restriction of RDF for DL-semantics.

A correct OWL axiom is: $Car \sqsubseteq \exists hasPart.Motor$

(c) If $\mathcal{I}$ satisfies the two triples, then there is a blank node assignment β such that $\langle \mathtt{x}^{\mathcal{I}}, \beta(\mathtt{\_:b}) \rangle \in \mathtt{p}^{\mathcal{I}}$ and $\beta(\mathtt{\_:b}) \in \mathtt{C}^{\mathcal{I}}$. We define $y := \beta(\mathtt{\_:b})$. Since $\langle \mathtt{x}^{\mathcal{I}}, y \rangle \in \mathtt{p}^{\mathcal{I}}$ and $y \in \mathtt{C}^{\mathcal{I}}$, it follows that $\mathtt{x}^{\mathcal{I}} \in (\exists \mathtt{p.C})^{\mathcal{I}}$.

If $\mathcal{I} \models \exists \mathtt{p.C}$, then there must be a $y \in C^{\mathcal{I}}$ with $\langle \mathtt{x}^{\mathcal{I}}, y \rangle \in \mathtt{p}^{\mathcal{I}}$. Choose a blank node assignment β with $\beta(\mathtt{\_:b}) = y$. Then $\mathcal{I}, \beta$ satisfy both triples.

(d)

$$\begin{aligned} \Delta^{\mathcal{I}} &= \{c\} \\ Car^{\mathcal{I}} &= \{c\} \\ Machinery^{\mathcal{I}} &= \emptyset \\ motor^{\mathcal{I}} &= \emptyset \end{aligned}$$

(1) is satisfied, because c is the only element of $Car^{\mathcal{I}}$, and does not have a *motor*-successor. Therefore, any *motor*-successor (namely none) is in $Machinery^{\mathcal{I}}$.

(2) is not satisfied, because c is in car, but it does not have a *motor*-successor, in particular not one which is in $Machinery^{\mathcal{I}}$.