UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580 — Semantic Technologies
Day of examination:	13 June 2012
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 dbp: <http://dbpedia.org/resource/> .
@prefix dbp-owl: <http://dbpedia.org/ontology/> .
@prefix dbp-prop: <http://dbpedia.org/property/> .

_:x a dbp-owl:PopulatedPlace ;
     rdfs:label "Oslo"@no ;
     dbp-owl:country _:y .

_:y a dbp-owl:PopulatedPlace ;
     rdfs:label "Norge"@no, "Norway"@en ;
     dbp-prop:areaKm "385252"^^xsd:integer ;
     dbp-owl:capital _:x .

dbp:UiO dbp-owl:country _:y;
     dbp-owl:city _:x .
```

Answer the following questions:

(Continued on page 2.)

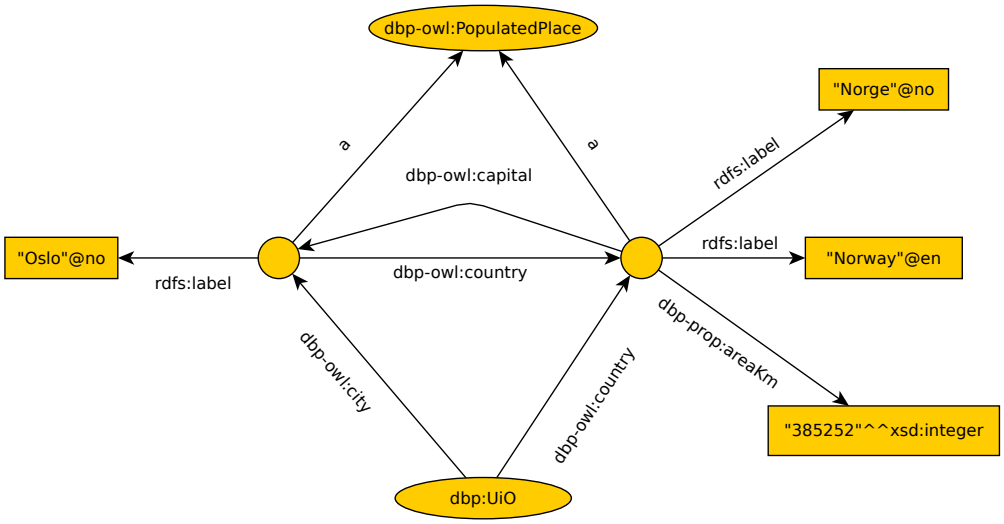
(a) Draw a graph representation of this RDF document.

(b) Explain briefly the difference between `dbp:Oslo` and `"Oslo"@no`.

(c) Add statements using the given `dbp-owl` properties to say that there is a *populated place* which has the *label* "Bergen", and which lies in the country with label "Norway" already mentioned.

Answer:

(a)



(b) `dbp:Oslo` denotes a resource, the intension is that it denotes the city of Oslo. `"Oslo"@no` denotes the (Norwegian) name of the city, i.e. a string of characters and not a city.

(c)
```
[a dbp-owl:PopulatedPlace;
 rdfs:label "Bergen";
 dbp-owl:country _:y]
```

Problem 2 SPARQL (20 %)

Consider an RDF document that contains information about airlines, airplanes, and flights.

The data uses the following classes in the `fly:` namespace:

Airport an airport, which can be origin or target of a connection

Aircraft a particular aircraft

(Continued on page 3.)

Airline a company that owns aircraft and operates flights

Connection a connection between two airports that has a "flight number" and that is usually flown regularly, e.g. once per day or once per week.

Flight a particular flight, which is some connection flown by some aircraft on a particular day.

and the following properties:

rdfs:label links airlines, airports, and aircraft to their names

fly:airline links connections to the airline that operates them, **and** aircraft to the airline that owns them.

fly:origin links a connection to the airport where it starts

fly:target links a connection to the airport where it ends

fly:flightNr links a connection to its flight number

fly:distanceKm links a connection to the distance between origin and target, in Km.

fly:type links aircraft to their type, i.e. producer and model (as a literal for simplicity)

fly:connection links a flight to the connection that is flown

fly:plane links a flight to the plane that is being used

fly:date links a flight to the day on which it starts

Here is some example data for *one* particular flight:

```
@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 fly:    <http://example.org/fly#> .

fly:Norwegian a fly:Airline;
    rdfs:label "Norwegian".

fly:OSL a fly:Airport;
    rdfs:label "Oslo, Gardermoen".

fly:CPH a fly:Airport;
    rdfs:label "Copenhagen, Kastrup".
```

(Continued on page 4.)

```
fly:DY940 a fly:Connection;
    fly:airline fly:Norwegian;
    fly:origin fly:OSL;
    fly:target fly:CPH;
    fly:flightNr "DY940";
    fly:distanceKm "518"^^xsd:double.

fly:LN-DYT a fly:Aircraft;
    rdfs:label "LN-DYT";
    fly:airline fly:Norwegian;
    fly:type "Boeing 737-8JP".

_:1 a fly:Flight;
    fly:connection fly:DY940 ;
    fly:plane fly:LN-DYT ;
    fly:date "2012-05-25"^^xsd:date .
```

Imagine that we have data about many more airlines, airports, aircraft, connections, and flights. 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 flight numbers of all connections operated by an airline called "Aeroflot".

(b) Write a query that lists the names of all airlines that operate connections with distance 5000km and more.

(c) The same flight number should not be used twice on the same day. Write a query that finds a flight number and date such that two *different* flights with that flight number go on that date.

(d) Sometimes, the aircraft that flies a connection is not owned by the same airline that operates the connection. E.g. a SAS connection might be flown by a plane that belongs to Lufthansa. Write a query that lists flight number and date and the names of the airline operating the connection and the airline owning the aircraft, for flights where these are not the same airline.

(e) Write a query that lists the name of every airline and the number of connections it operates which have a flight on 13 June 2012 (`"2012-06-13"^^xsd:date`)

Answer:

(Continued on page 5.)

(a)
```
SELECT ?fnr WHERE {
  [] a fly:Connection ;
     fly:airline [rdfs:label "Aeroflot"] ;
     fly:flightNr ?fnr .
}
```

(b)
```
SELECT DISTINCT ?name WHERE {
  [] a fly:Connection ;
     fly:airline [rdfs:label ?name] ;
     fly:distanceKm ?d .
  FILTER (?d > 5000)
}
```

(c)
```
SELECT DISTINCT ?fnr ?date WHERE {
  ?f1 a fly:Flight ;
      fly:connection [fly:flightNr ?fnr] ;
      fly:date ?date .
  ?f2 a fly:Flight ;
      fly:connection [fly:flightNr ?fnr] ;
      fly:date ?date .
  FILTER ( ?f1 != ?f2 )
}
```

(d)
```
SELECT ?fnr ?date ?op ?own WHERE {
  [] a fly:Flight ;
     fly:date ?date ;
     fly:plane [fly:airline [rdfs:label ?own]] ;
     fly:connection [fly:airline [rdfs:label ?op];
                     fly:flightNr ?fnr] .
  FILTER ( ?own != ?op)
}
```

(e)
```
SELECT ?name (COUNT(?c) as ?count) {
  [] a fly:Flight ;
     fly:date "2012-06-13"^^xsd:date ;
     fly:connection ?c .
  ?c fly:airline [rdfs:label ?name] .
} GROUP BY ?name
```

Problem 3 RDFS Reasoning (20 %)

Let the following set of triples be given:

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
```

(Continued on page 6.)

```
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix v: <http://example.org/vocab/> .
@prefix f: <http://example.org/oilfields/> .
@prefix c: <http://example.org/companies/> .
```

(1) `v:hasPartner rdfs:domain v:Oilfield .`

(2) `v:hasPartner rdfs:range v:Company .`

(3) `v:hasOperator rdfs:subPropertyOf v:hasPartner .`

(4) `v:hasNonOperatingPartner rdfs:subPropertyOf v:hasPartner .`

(5) `v:OperatingCompany rdfs:subClassOf v:Company .`

(6) `v:operates rdfs:domain v:OperatingCompany .`

(7) `v:operates rdfs:range v:Oilfield .`

(8) `f:ekofisk v:hasOperator c:conoco .`

(9) `f:ekofisk v:hasNonOperatingPartner c:statoil .`

(10) `f:troll v:hasOperator c:statoil .`

For each of the following triples (or sets of triples, in (e)), 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, also indicate whether the statement is entailed or not (under the simplified RDF/RDFS semantics used in the course).

(a) `f:ekofisk v:hasPartner c:statoil`

(b) `c:conoco a v:Company`

(c) `c:conoco a v:OperatingCompany`

(d) `v:hasOperator rdfs:domain v:Company`

(e)
```
_:x v:hasOperator _:y .
_:z v:hasNonOperatingPartner _:y .
```

Answer:

(a) (a1) `f:ekofisk v:hasPartner c:statoil` from (4) and (9) by rdfs7

(b) (b1) `f:ekofisk v:hasPartner c:conoco` from (3) and (8) by rdfs7
(b2) `c:conoco a v:Company` from (b1) and (2) by rdfs3

(Continued on page 7.)

(c) This cannot be derived and it isn't entailed. We could derive it using (6) if we knew e.g. `c:conoco v:operates f:ekofisk`. But RDFS cannot express that `v:operates` is the inverse of `v:hasOperator`.

(d) This cannot be derived, since there is no rule to derive new `rdfs:domain` statements. It isn't entailed either (in fact, it is entailed that `v:hasOperator rdfs:range v:Company`)

(e) (e1) `_:x v:hasOperator c:statoil` from (10) by se2, allocating `_:x` to `f:troll`

(e2) `_:x v:hasOperator _:y` from (e1) by se1, allocating `_:y` to `c:statoil`

(e3) `_:z v:hasNonOperatingPartner c:statoil` from (9) by se2, allocating `_:z` to `f:ekofisk`

(e4) `_:z v:hasNonOperatingPartner _:y` from (e3) by se1, reusing the allocation of `_:y` to `c:statoil`

Triples (e2) and (e4) are the ones that had to be derived.

Problem 4 Description logics/OWL (20 %)

Consider information about wines, wine growing regions, and producers represented in the following way:

```
wine:mr1982 a :Wine ;
      :name "Château Mouton Rothschild 1982" ;
      :appelation :pauillac ;
      :vintage "1982"^^xsd:int ;
      :producer :mr .

:pauillac a :Appelation ;
      :containedIn :hautMedoc .

:hautMedoc a :SubRegion ;
      :containedIn :bordeaux .

:bordeaux a :Region ;
      :containedIn :france .

:france a :Country .

:mr a :Producer ;
      :produces wine:mr1982 .
```

(Continued on page 8.)

Note: An "appelation" is printed on the label to say where the wine comes from. Pauillac in the example is a town, which lies in Haut-Médoc, which lies in Bordeaux, etc.

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: Wine, Appelation, SubRegion, Region, Country, Producer
- Properties: name, appelation, vintage, producer, produces, containedIn

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

(a) Every producer produces at least one wine

(b) Every wine has exactly one producer

(c) From the *containedIn*-statements in the example, we would like to be able to infer, e.g., that Pauillac is also contained in Haut-Médoc, Bordeaux, France, etc. I.e. if *a* is *containedIn b* and *b* in *c*, then *a* is *containedIn c*. How can this be required using OWL?

(d) If a wine *w* has a *:producer p*, then *p :produces* that wine *w*, and vice versa.

(e) Wine was not produced before 8000 BC, i.e. every wine has a vintage after -8000.

(f) A GrandOldProducer is defined to be a producer who produces at least five wines with vintage 1980 or older.

Answer:

(a) $Producer \sqsubseteq \exists produces.Wine$
Producer **SubClassOf** produces **some** Wine

(b) $Wine \sqsubseteq =_1 producer.Producer$
Wine **SubClassOf** producer **exactly** 1 Producer

(c) containedIn has to be declared to be *transitive*. DL syntax varies. Manchester syntax:

```
ObjectProperty: containedIn
  Characteristics: Transitive
```

(d) `producer` and `produces` need to be declared *inverses* of each other. DL syntax varies. Manchester syntax:

(Continued on page 9.)

```
ObjectProperty: producer
  InverseOf: produces
```

(e) $Wine \sqsubseteq \exists vintage.int[>= -8000]$
Wine **SubClassOf** vintage some int [>= -8000]

(f) $GrandOldProducer \equiv Producer \sqcap \geq_5 produces.(Wine \sqcap \exists vintage.int[<= 1980])$
GrandOldProducer **EquivalentTo** Producer **and** produces **min** 5
(Wine **and** vintage **some** int [<= 1980]])

Problem 5 RDF and OWL semantics (20 %)

(a) Let C and D be classes, and R an object property. The following axiom is valid in every interpretation:

$$\forall R.(C \sqcap D) \equiv (\forall R.C) \sqcap (\forall R.D)$$

Give a short proof sketch of this fact.

(b) Now consider:

$$\forall R.(C \sqcup D) \equiv (\forall R.C) \sqcup (\forall R.D)$$

Is this valid in all interpretations? If yes, give a proof sketch, if no, give an interpretation (consisting of $\Delta^{\mathcal{I}}$, $C^{\mathcal{I}}$, $D^{\mathcal{I}}$, and $R^{\mathcal{I}}$) where it is not valid.

(c) Let R be a datatype property. Give a DL-interpretation $\mathcal{I}$, consisting of $\Delta^{\mathcal{I}}$, $C^{\mathcal{I}}$, $D^{\mathcal{I}}$, and $R^{\mathcal{I}}$, that satisfies the following axioms:

- C **EquivalentTo** R **min** 2
  or in DL syntax: $C \equiv \geq_2 R$
- D **EquivalentTo** R **some int** [>= 2]
  or in DL syntax: $D \equiv \exists R.[>= 2]$

such that each of $(C \sqcap D)^{\mathcal{I}}$, $(C \sqcap \neg D)^{\mathcal{I}}$, $(\neg C \sqcap D)^{\mathcal{I}}$, and $(\neg C \sqcap \neg D)^{\mathcal{I}}$ have at least one element.

Answer:

(a) Let $\mathcal{I}$ be an interpretation. Then

$$\begin{aligned}
(\forall R.(C \sqcap D))^{\mathcal{I}} &= \{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \text{ with } xR^{\mathcal{I}}y,\ y \in C^{\mathcal{I}} \text{ and } y \in D^{\mathcal{I}}\} \\
&= \{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \text{ with } xR^{\mathcal{I}}y,\ y \in C^{\mathcal{I}}\} \\
&\quad \cap \{x \in \Delta^{\mathcal{I}} \mid \text{for all } y \text{ with } xR^{\mathcal{I}}y,\ y \in D^{\mathcal{I}}\} \\
&= (\forall R.C)^{\mathcal{I}} \cap (\forall R.D)^{\mathcal{I}} \\
&= (\forall R.C \sqcap \forall R.D)^{\mathcal{I}}
\end{aligned}$$

(Continued on page 10.)

(b) Define

$$\Delta^{\mathcal{I}} = \{a, c, d\}$$
$$C^{\mathcal{I}} = \{c\}$$
$$D^{\mathcal{I}} = \{d\}$$
$$R^{\mathcal{I}} = \{(a, c), (a, d)\}$$

Then

$$(\forall R.(C \sqcup D))^{\mathcal{I}} = \{a, c, d\}$$

because every R-successor of a is *either* in C *or* in D. (c and d are included because they don't have R-successors) On the other hand,

$$(\forall R.C)^{\mathcal{I}} = (\forall R.D)^{\mathcal{I}} = \{c, d\}$$

because the R-successors of a are neither all in C nor all in D.

(c) C is the class of objects having at least two different values for R, while D is the class of objects having at least one value for R that is larger than 2.

We choose a domain $\Delta^{\mathcal{I}} = \{o, c, d, cd\}$ and

$$C = \{c, cd\} \qquad D = \{d, cd\}$$

which implies that $cd \in (C \sqcap D)^{\mathcal{I}}$, $c \in (C \sqcap \neg D)^{\mathcal{I}}$, $d \in (\neg C \sqcap D)^{\mathcal{I}}$, and $o \in (\neg C \sqcap \neg D)^{\mathcal{I}}$.

To satisfy the axioms, we can define for instance:

$$R^{\mathcal{I}} = \{(c, 0), (c, 1), (d, 2), (cd, 1), (cd, 2)\}$$