UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580/INF4580 — Semantic Technologies
Day of examination:	4 June 2015
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 5 questions with equal weight.

Problem 1 RDF (20 %)

The Norwegian Business Registry in Brønnøysund keeps information about "units" such as companies, organisations, etc. that are active in Norway. Every unit in the register is uniquely identified by a 9 digit "organisation number." The dataset is very large, it contains over 1 million units.

Assume that the Brønnøysund registry now wants to make the information about these units available as RDF data.

(a) There has to be a URI for every unit, and data should be published using linked open data (LOD) principles, such that the URI for a unit can be used to retrieve basic information about that unit. Which of the following suggestions for the URI is better, and why?

1. `http://brreg.no/enheter/`⟨*org.nr.*⟩
2. `http://brreg.no/enheter#`⟨*org.nr.*⟩

(b) What is "HTTP 303" redirection, and what is its purpose in the context of LOD?

(c) Consider the following HTML + RDFa 1.0 file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
```

(Continued on page 2.)

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:v="http://rdf.data-vocabulary.org/#"
      xmlns:brreg="http://brreg.no/vocab#">
  <head>
    <title>UNIVERSITETET I OSLO</title>
  </head>

  <body about="http://brreg.no/enheter/971035854" typeof="brreg:Enhet">

    <h1>Information about
      <span property="v:name">UNIVERSITETET I OSLO</span></h1>

    <p>Organisation Number
      <span property="brreg:orgNr">971035854</span></p>

    <p>Address:</p>
    <div rel="v:address">
      <address typeof="v:Address">
<span property="v:street-address">Postboks 1072 Blindern</span><br/>
<span property="v:postal-code">0316</span>
<span property="v:locality">OSLO</span>
      </address>
    </div>
  </body>
</html>
```

Please write the triples included in this file in Turtle format.

Answer:

(a) Variant 1, with the slash namespace is more sensible. The reason is that the "fragment identifier" after the hash sign has to be stripped from a URL before retrieving information with HTTP. Accessing the particular fragment is up to the client. In this example, a client would have to access `http://brreg.no/enheter`, which presumably contains information about all the one million units.

(b) The status code "303 See Other" is a possible reaction of an HTTP server to a request. It means that it will not serve the resource asked for but instructs the client to access a different URL. That other URL will be given in the "Location:" response header.

In the context of LOD, 303 redirection is used to separate resources from documents describing the resources, while at the same time keeping dereferencable URIs for non-document resources. Accessing a URI like `http://www.dbpedia.org/resource/Oslo` which denotes a city, the server responds with a 303 redirection to an HTML page file, an RDF file, etc., at a different location, that contains information about Oslo.

(c) `@prefix brreg: <http://brreg.no/vocab#> .`

(Continued on page 3.)

```
@prefix v: <http://rdf.data-vocabulary.org/#> .

<http://brreg.no/enheter/971035854> a brreg:Enhet;
   brreg:orgNr "971035854";
   v:address [ a v:Address;
         v:locality "OSLO";
         v:postal-code "0316";
         v:street-address "Postboks 1072 Blindern" ];
   v:name "UNIVERSITETET I OSLO" .
```

Problem 2 SPARQL (20 %)

Assume we have the two vocabularies wth and mov that describe weather and movies at movie theatres respectively. In wth we have the following defined URIs:

wth:DailyForecast (class) is the class of daily forecasts.

wth:dateTime (data property) gives the date as an xsd:dateTime string of the form "yyyy-mm-dd" for a a forecast.

wth:forecast (data property) states the actual weather forecast as a string for a given forecast. The forecast can be one of the literals "clear", "cloudy", "light rain", and "heavy rain".

In the mov vocabulary, we have the following URIs:

mov:Movie (class) the movies that can be shown in movie theatres.

mov:MovieTheatre (class) the movie theatres that show movies.

mov:Screening (class) is all movie screenings. A movie screening is one display of a movie at a specific theatre at a specific date.

mov:theatre (object property) relates a movie screening to the theatre it runs at.

mov:shownMovie (object property) the movie shown in a screening.

mov:dateTime (data property) the date of a screening, which is an xsd:dateTime string on the form "yyyy-mm-dd".

mov:name (data property) relates movies to their title and movie theatres to their names.

Here are some example triples of the above vocabulary:

(Continued on page 4.)

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
@prefix wht: <http://www.ifi.uio.no/weather#>
@prefix mov: <http://www.ifi.uio.no/movies#>

wht:fc1 a wht:DailyForecast;
  wth:dateTime "2015-05-07"^^xsd:dateTime;
  wht:forecast "light rain" .

mov:hobbitScreening a mov:Screening;
  mov:dateTime "2015-01-02"^^xsd:dateTime;
  mov:theater mov:colosseum;
  mov:shownMovie mov:hobbit .

mov:colosseum a mov:MovieTheater;
  mov:name "Colosseum" .

mov:hobbit a mov:Movie;
  mov:name "The Hobbit: The Battle of the Five Armies" .
```

(a) Write a SPARQL query that retrieves all dates where the forecast is cloudy .

(b) Write a SPARQL query returns *true* if there exists a date after 1. January 2015 where the forecast is cloudy and *false* otherwise (you can assume that `xsd:dateTime` is ordered chronologically by (<)).

(c) Assume that the weather station is somewhat unstable in your area, and that there might be days where there is no forecast. Write a SPARQL query that lists all movies on dates where, there is no forecast, or *if* there is a forecast, it should be heavy rain.

(d) Write a SPARQL query that finds all dates where there is a screening of a movie that has a title that starts with the string “The Hobbit:” or where the forecast is clear.

(e) Write a SPARQL query that finds the theatres that have more than 5 screenings on a rainy day (either light or heavy rain). The query should output both the theatre, the date and the number of screenings.

Answer:

(a)
```
PREFIX wth: <http://www.ifi.uio.no/weather#>
PREFIX mov: <http://www.ifi.uio.no/movies#>

SELECT ?dateStr
WHERE {
```

(Continued on page 5.)

```
  ?forecast a wth:DailyForecast;
            wth:dateTime ?dateStr;
            wth:forecast "cloudy" .
}
```

(b)
```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX wth: <http://www.ifi.uio.no/weather#>

ASK
WHERE {
  ?forecast a wth:DailyForecast;
            wth:dateTime ?dateStr;
            wth:forecast "cloudy" .
  FILTER (?dateStr >= "2015-01-01"^^xsd:dateTime)
}
```

(c)
```
PREFIX wth: <http://www.ifi.uio.no/weather#>
PREFIX mov: <http://www.ifi.uio.no/movies#>

SELECT ?title WHERE {
   ?screening a mov:Screening;
      mov:movie [a :mov:Movie; mov:name ?title];
      mov:dateTime ?dateStr .

   FILTER NOT EXISTS {
      [] a wth:DailyForecast;
         wth:dateTime ?dateStr ;
         wth:forecast ?fc .
      FILTER (?fc != "heavy rain")
   }
}
```

eller alternativt

(d)
```
PREFIX wth: <http://www.ifi.uio.no/weather#>
PREFIX mov: <http://www.ifi.uio.no/movies#>

SELECT ?dateStr
WHERE {
  {
    ?screening a mov:Screening;
               mov:dateTime ?dateStr;
               mov:movie ?movie .
    ?movie a mov:Movie;
           mov:name ?name .
    FILTER (regex(str(?name), "The Hobbit:"))
```

(Continued on page 6.)

```
  }
  UNION
  {
    ?forecast a wth:DailyForecast;
              wth:dateTime ?dateStr;
              wth:forecast "clear" .
  }
}
```

(e)

```
PREFIX wth: <http://www.ifi.uio.no/weather#>
PREFIX mov: <http://www.ifi.uio.no/movies#>

SELECT ?theatreStr ?dateStr (count(?screening) AS ?sum)
WHERE {
  ?screening a mov:Screening;
             mov:theatre ?theatre;
             mov:dateTime ?dateStr .
  ?theatre a mov:Theatre;
           mov:name ?theatreStr .
  ?forecast a wth:DailyForecast;
            wth:dateTime ?dateStr;
            wth:forecast ?fcStr .
  FILTER (?fcStr = "heavy rain" || ?fcStr = "light rain")
} HAVING (?sum > 5)
```

Problem 3 RDFS Reasoning (20 %)

Consider the following triples about movies using the same vocabulary as the previous Problem:

```
    @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
    @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
    @prefix foaf:<http://xmlns.com/foaf/0.1/> .
    @prefix p: <http://example.org/people/> .
    @prefix : <http://www.ifi.uio.no/movies#> .
(1) :hasDirector rdfs:domain :Movie .
(2) :hasActor rdfs:range :Actor .
(3) :hasMaleLead rdfs:range :Man .
(4) :hasMaleLead rdfs:subPropertyOf :hasActor .
(5) :hasFemaleLead rdfs:range :Woman .
```

(Continued on page 7.)

(6) `:hasFemaleLead rdfs:subPropertyOf :hasActor .`

(7) `:Actor rdfs:subClassOf :Person .`

(8) `:hobbit :hasDirector _:1 .`

(9) `_:1 foaf:name "Peter Jackson" .`

(10) `:hobbit :hasMaleLead p:martin .`

(11) `p:martin foaf:name "Martin Freeman" .`

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) `:hobbit a :Movie .`

(b) `p:martin a :Person .`

(c) `:hasMaleLead rdfs:range :Person .`

(d) `:hasFemaleLead rdfs:domain :Movie .`

(e) `_:x :hasActor _:y .`
`_:x :hasDirector _:y .`

Answer: The rule applications refer to the rule names found in `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/#RDFSRules`.

(a)

(a1) `:hobbit a :Movie` (by rule rdfs2 on (1), (8))

(b)

(b1) `:hobbit :hasActor p:martin` (by rule rdfs7 on (4), (10))

(b2) `p:martin a :Actor` (by rule rdfs3 on (2), (b1))

(b3) `p:martin a :Person` (by rule rdfs9 on (7), (b2))

(c) No RDFS entailment rule lets one add triples where `rdfs:range` is the predicate.

The triple is entailed under the simplified RDFS semantics however: if in any interpretation, *x* `:hasMaleLead` *y*, then *x* `:hasActor` *y*, due to (4), and therefore *y* `rdf:type :Actor` by (2). Then, due to (7), we also have *y* `rdf:type :Person`. Together, this means that `:hasMaleLead rdfs:range :Person` is entailed.

This is an example of the *incompleteness* of the RDFS rule set.

(Continued on page 8.)

(d) As before, no RDFS entailment rule can be used to add triples with predicate `rdfs:domain`.

The triple is not even entailed: nothing is known about the domain of `:hasFemaleLead` or any of its superproperties.

(e) These triples are not entailed. For an interpretation and blank node assignment to make them true, the blank node assignment would have to assign the same individual to both occurrences of `_:x` and likewise for `_:y`. In other words, one would need a movie where one person was both actor and director, and there is no such movie in this dataset.

Note in particular: there are interpretations of the dataset where the blank node `_:1` is assigned to the same individual as `p:martin`. But for entailment, that would have to hold in *all* interpretations—which is not the case.

Problem 4 Description logics/OWL (20 %)

Assume we have the following classes:

Postman, Route, Mail, PriorityMail, RegularMail

the following properties:

hasRoute, containsAddress, deliversToAddress, mailTo

and the following individuals:

king, pat

denoting the King and the person Pat respectively.

The property hasRoute is a relation between postmen and their routes, containsAddress is a relation between routes and the addresses on that route, deliversToAddress is a relation between postmen and the addresses they deliver mail to, and mailTo is a relationship between mail and the address the mail is for.

Express the following sentences as DL axioms:

(a) All postmen have a route.

(b) Pat is not a postman.

(c) All mail is either priority mail or regular mail, but never both.

(d) Postmen have between one and five routes.

(Continued on page 9.)

(e) If something/someone has a route and that route contains an address, then that something/someone delivers mail to that address.

(f) All mail to the King is priority mail.

Answer:

(a) Postman $\sqsubseteq \exists$hasRoute.$\top$.

(b) {pat} $\sqsubseteq \neg$Postman.

(c) Mail $\sqsubseteq$ PriorityMail $\sqcup$ RegularMail and RegularMail $\sqcap$ PriorityMail $\sqsubseteq \bot$.

(d) Postman $\sqsubseteq \geq_1$ hasRoute.$\top \sqcap \leq_5$ hasRoute.$\top$.

(e) hasRoute $\circ$ containsAddress $\sqsubseteq$ deliversToAddress .

(f) $\exists$mailTo . {king} $\sqsubseteq$ PriorityMail. Also $\forall$mailTo . {king} $\sqsubseteq$ PriorityMail is correct, if we assume that all mail is at least to one person. Another variant: {king} $\sqsubseteq \forall mailTo^{-1}$.PriorityMail

Problem 5 RDF and OWL semantics (20 %)

In this problem, A, B, C and D are class names, a and b are individual names, and R is a role name.

(a) Let $\mathcal{I}$ be a DL-interpretation with $\mathcal{I} \models C \sqsubseteq D$. Show that also $\mathcal{I} \models \forall R.C \sqsubseteq \forall R.D$.

Note: This fact can be useful for the following questions!

(b) Given the following set $\mathcal{T}$ of TBox axioms:

$$\mathcal{T} = \{ \begin{array}{rl} A \sqcap B & \sqsubseteq \bot, \\ B & \sqsubseteq \forall R.\bot, \\ \forall R.D & \sqsubseteq A \end{array} \}$$

Are the axioms in $\mathcal{T}$ consistent with the ABox assertion $A(a)$? If yes, give an interpretation satisfying the three axioms and the assertion. If no, give a short explanation why, based on DL semantics.

(c) Are the axioms in $\mathcal{T}$ consistent with the ABox assertion $B(b)$? If yes, give an interpretation satisfying the three axioms and the assertion. If no, give a short explanation why, based on DL semantics.

Answer:

(Continued on page 10.)

(a) Let $x \in \Delta^{\mathcal{I}}$ with $x \in (\forall R.C)^{\mathcal{I}}$. Then for all y with $\langle x, y \rangle \in R^{\mathcal{I}}$, $y \in C^{\mathcal{I}}$. Since $\mathcal{I} \models C \sqsubseteq D$, we also have $y \in D^{\mathcal{I}}$, and therefore $x \in (\forall R.D)^{\mathcal{I}}$.

(b) Yes, this is consistent. We can pick an interpretation with $\Delta^{\mathcal{I}} = \{1\}$. Interpret for instance $a^{\mathcal{I}} = 1$, $A^{\mathcal{I}} = \{1\}$, $B^{\mathcal{I}} = C^{\mathcal{I}} = D^{\mathcal{I}} = \emptyset$ and $R^{\mathcal{I}} = \emptyset$.

(c) No, this is not consistent. For assume that we have an interpretation $\mathcal{I}$ that makes all axioms and the assertion true. Let $x = b^{\mathcal{I}}$. Then due to the ABox assertion $B(b)$, $x \in B^{\mathcal{I}}$. Due to the second axiom, $x \in (\forall R.\bot)^{\mathcal{I}}$. Since $\bot \sqsubseteq D$ is true in any interpretation, together with part (a), we get $x \in (\forall R.D)^{\mathcal{I}}$. The third axiom then gives $x \in A^{\mathcal{I}}$. Together, $x \in A^{\mathcal{I}} \cap B^{\mathcal{I}} = (A \sqcap B)^{\mathcal{I}}$, which contradicts the first axiom.