UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3580/INF4580 — Semantic Technologies
Day of examination:	28 May 2014
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 %)

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 b: <http://www.ifi.uio.no/bones#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

b:KneeBone rdfs:subClassOf b:Bone .
b:ShinBone rdfs:subClassOf b:Bone .

b:martin a foaf:Person ;
         b:hasPart _:a,_:b .
_:a a b:ShinBone ;
    b:connectedTo _:b .
_:b a b:KneeBone ;
    b:connectedTo _:a .
```

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

(b) Using the vocabulary in the graph and an additional class `b:Leg`, express the following statement:

(Continued on page 2.)

Martin has a leg. This leg has the shin bone and knee bone identified respectively by the blank node identifiers `_:a` and `_:b` as parts.

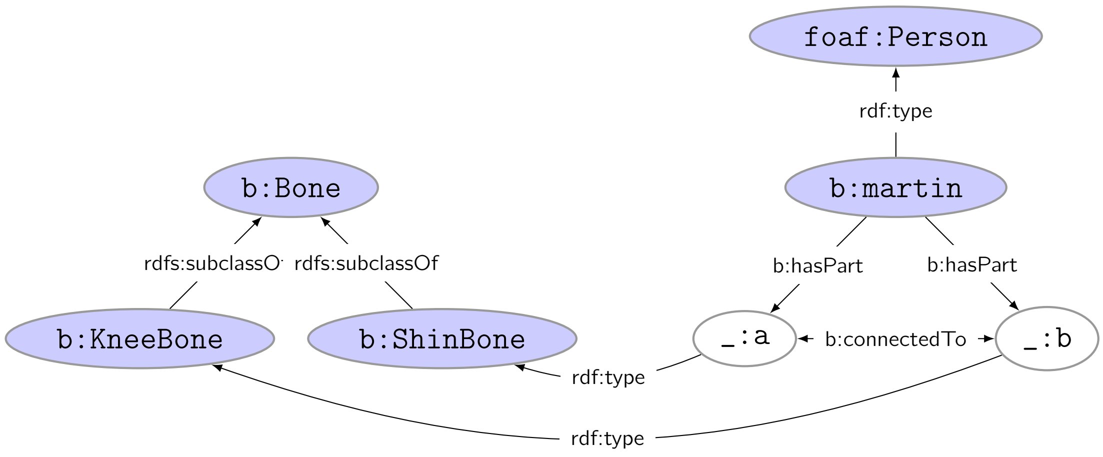
(c) Is it possible to represent the same RDF graph in Turtle, without the use of blank node identifiers like `_:a` and `_:b`? Show how, or briefly explain why not.

(d) Decide if the following statement is correct or wrong, and briefly explain why: "Using URIs to identify resources in RDF ensures that every resource will have only one identifier. Two different URIs will always refer to two different things."

(e) Decide if the following statement is correct or wrong, and briefly explain why: "Using URIs to identify resources in RDF ensures that every identifier only refers to one resource. The same URI cannot refer to two different things."

Answer:

(a)



(b)
```
b:martin b:hasPart [ a b:Leg ;
                     b:hasPart _a:, _:b ] .
```

(c) No. Without blank node identifiers is it not possible to represent the two triples

```
_:a connectedTo _:b .
_:b connectedTo _:a .
```

The `[...]` syntax can be used to refer to 'new' blank nodes, but there is no way to refer back to a previously used blank node without resource identifiers.

(Continued on page 3.)

(d) Wrong. A resource can have more than one URI, e.g., `dbp:Norway`, `http://sws.geonames.org/3144096/` and `freebase:Norway` are all URIs identifying Norway, see, e.g., `http://dbpedia.org/page/Norway`.

(e) Wrong. Within one interpretation, a URI refers to exactly one domain element (= resource). But the same URI may refer to different domain elements in different interpretations.

In other words: it is not possible to force a URI to be interpreted as a certain resource from the 'real world,' the interpretation is always only given relative to some context.

Problem 2 SPARQL (20 %)

Assume that we are given an RDF database of the cars, drivers, driving licenses and deliveries of a delivery service.

The vocabulary used is the following, in addition to FOAF for people and their names.

dlv:Delivery (class) a delivery happening on some day, using a particular vehicle, and driver, going to a particular destination. We assume that all deliveries take only one day, and the whole day.

dlv:Vehicle (class) a vehicle that may be used in several deliveries.

dlv:date (property) the date on which a delivery is driven.

dlv:vehicle (property) the vehicle used for a delivery.

dlv:vehicleKMs (property) the kilometres the vehicle had driven at the beginning of this delivery.

dlv:driver (property) the vehicle used for a delivery.

dlv:destination (property) the vehicle used for a delivery.

dlv:registration (property) the registration number of a vehicle.

dlv:weightKG (property) the vehicle's weight in kg.

dlv:licenseClass (property) the driver's license class of a person.

Here are some example triples.

(Continued on page 4.)

```
@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 xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dlv: <http://www.ifi.uio.no/delivery#> .

[a dlv:Delivery;
   dlv:date "2014-05-17"^^xsd:dateTime;
   dlv:vehicle dlv:AV43634;
   dlv:vehicleKMs "123654"^^xsd:double ;
   dlv:driver dlv:BjarneBerg;
   dlv:destination dlv:Bergen
] .

dlv:AV43634 a dlv:Vehicle ;
   dlv:registration "AV43634"^^xsd:string ;
   dlv:weightKG "5000"^^xsd:double .

dlv:BjarneBerg a foaf:Person ;
   foaf:name "Bjarne Berg";
   dlv:licenseClass dlv:classB .
```

Imagine that we have data about many more deliveries, cars, and people.

(a) Write a query that lists the registration numbers (without repetitions) of the vehicles driven by a person with name Magne Mo.

(b) Write a query that finds cases where a driver with a class B license (see example triples) has driven a vehicle weighing more than 3500kg. List the name of the driver, the registration number, and the date.

(c) Write a query that finds cases where two different vehicles have a delivery to the same destination on the same day. List the date and the two registration numbers.

(d) Write a query that finds cases where the number of driven kilometres for one vehicle goes *down* from one date to a later date. List the registration number and the two dates and driven distances.

(e) For each vehicle, list the registration number and the number of kilometres driven in April 2014. You can assume that every time a vehicle is used, this is recorded in your data. Hint: use SPARQL 1.1 sub-queries to find the first delivery on 1 April or later (for the driven km at the beginning of April), and the first delivery after 31 April (for the driven km at the end of April)

(Continued on page 5.)

Answer:

NOTE: there were some typos in the questions. The descriptions of dlv:driver and dlv:destination were wrong. The intended meaning should be obvious from the names and the example data.

(a)
```
SELECT DISTINCT ?regno {
  [] a dlv:Delivery ;
     dlv:vehicle [a dlv:Vehicle; dlv:registration ?regno ] ;
     dlv:driver  [a foaf:Person; foaf:name "Magne Mo" ] .
}
```

(b)
```
SELECT DISTINCT ?name ?regno ?date  {
  [] a dlv:Delivery ;
     dlv:date ?date ;
     dlv:vehicle [a dlv:Vehicle ;
                    dlv:registration ?regno ;
                    dlv:weightKG ?weight ] ;
     dlv:driver [ a foaf:Person;
                  dlv:licenceClass dlv:classB;
                  foaf:name ?name ] .
  FILTER (?weight > 3500)
}
```

(c)
```
SELECT DISTINCT ?date ?regA ?regB {
  [] a dlv:Delivery;
     dlv:date ?date ;
     dlv:destination ?dest ;
     dlv:vehicle [ a dlv:Vehicle;
                   dlv:registration ?regA ] .
  [] a dlv:Delivery;
     dlv:date ?date ;
     dlv:destination ?dest ;
     dlv:vehicle [ a dlv:Vehicle;
                   dlv:registration ?regB ] .

  FILTER (?regA != ?regB)
}
```

Can filter in a number of ways, either make sure the two deliveries are different (since a delivery is assumed to last a whole day, and only have one vehicle, this would be enough), or the two vehicles (URIs) are different, or, like we do here, we accept filtering on registration numbers, which can be assumed to identify a vehicle.

(d)
```
SELECT DISTINCT ?reg ?dateA ?dateB ?distA ?distB {
  [] a dlv:Delivery;
```

(Continued on page 6.)

```
      dlv:date ?dateA ;
      dlv:vehicleKM ?distA ;
      dlv:vehicle ?v .
   [] a dlv:Delivery;
      dlv:date ?dateB ;
      dlv:vehicleKM ?distB ;
      dlv:vehicle ?v .
   ?v a dlv:Vehicle;
      dlv:registration ?reg .
   FILTER (?dateA < ?dateB &&
           ?distA > ?distB)
}
```

This will find every combination of dates where the condition holds, and not the closest dates, but that was not asked for. Can also use the reg. number to make sure the two deliveries are for the same vehicle.

(e)
```
SELECT ?reg ?dist {
   ?v a dlv:Vehicle;
      dlv:registration ?reg .

   # find the first delivery in April:
   SELECT ?v ?kmA {
     []  a dlv:Delivery;
           dlv:date ?dateA ;
           dlv:vehicle ?v ;
           dlv:vehicleKM ?kmA ;
     FILTER (?dateA >= "2014-04-01"^^xsd:dateTime)
     ORDER BY ASC(?dateA) LIMIT 1;
   }

   # find the first delivery in May:
   SELECT ?v ?kmB{
     []  a dlv:Delivery;
           dlv:date ?dateB ;
           dlv:vehicle ?v ;
           dlv:vehicleKM ?kmB ;
     FILTER (?dateB >= "2014-05-01"^^xsd:dateTime)
     ORDER BY ASC(?dateB) LIMIT 1;
   }

   # calculate distance:
   BIND (?kmB - ?kmA AS ?dist)
}
```

(Continued on page 7.)

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#> .
     @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
     @prefix d: <http://example.org/docs/> .
     @prefix a: <http://example.org/authors/> .
     @prefix v: <http://example.org/vocab/> .
 (1) v:inJournal rdfs:domain v:Article .
 (2) v:inJournal rdfs:range v:Journal .
 (3) v:Book rdfs:subClassOf v:Document .
 (4) v:Article rdfs:subClassOf v:Document .
 (5) v:hasFirstAuthor rdfs:subPropertyOf v:hasAuthor .
 (6) v:hasAuthor rdfs:range v:Author .
 (7) d:a v:hasFirstAuthor a:uli .
 (8) d:a v:hasAuthor a:ian .
 (9) d:a v:inJournal d:jws .
(10) d:a v:cites _:x .
(11) _:x a v:Book .
(12) _:x v:hasFirstAuthor a:franz .
(13) _:x v:cites d:a .
```

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) `d:a a v:Document`

(b) `a:franz a v:Author`

(c) `v:hasFirstAuthor rdfs:domain v:Document`

(d) `v:hasFirstAuthor rdfs:range v:Author`

(e)
```
_:x v:cites _:y .
_:y v:cites _:z .
_:z a v:Article .
```

(Continued on page 8.)

Answer:

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

(a)

(1) `d:a a v:Article` rdfs2 + (1) + (9)

(2) `d:a a v:Document` rdfs9 + (4) + (a1)

(b)

(1) `_:x v:hasAuthor a:franz` rdfs7 + (5) + (12)

(2) `a:franz a v:Author` rdfs3 + (b1) + (6)

(c) No RDFS entailment rule lets one add triples where `rdfs:domain` is the predicate. This triple is also not entailed under the semantics, as there are no axioms describing the domain of `v:hasFirstAuthor` or its super-property `v:hasAuthor`.

(d) No RDFS entailment rule lets one add triples where `rdfs:range` is the predicate. This triple is however entailed by (5) and (6) in in the simplified semantics we use in the course, and can be derived if we allow the use of the extended entailment rules of RDFS, in this case ext4.

(e) To check entailment, using the simple entailment rules, we first rename the variables in the entailed graph to `_:x1`, `_:y1`, and `_:z1`.

(1) `_:x1 v:cites _:x` se2 + (10) with `_:x1`→`d:a`

(2) `_:x1 v:cites _:y1` se1 + (e1) with `_:y1`→`_:x`

(3) `_:y1 v:cites d:a` se2 + (13) reusing `_:y1`→`_:x`

(4) `_:y1 v:cites _:z1` se1 + (e3) with `_:z1`→`d:a`

(5) `d:a a v:Article` rdfs2 + (1) + (9)

(6) `_:z1 a v:Article` se2 + (e5) reusing `_:z1`→`d:a`

The triples (2), (4), and (6) are the ones to be derived, after the renaming.

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: Company, Field, License, ActiveField, OperatorCompany

(Continued on page 9.)

- Properties: operator, partner, hasLicense, startDate, endDate

A "license" gives several companies the right to exploit the petroleum reserves of a "field." These companies are known as the "partners" of the license, and one (and only one) of those partners is known as the "operator."

So "hasLicense" is the object property that links a field to a license, and "operator" and "partner" are object properties that link the license to its operator company and partner companies.

A license is always valid for a certain period in time. We use the startDate and endDate datatype properties to indicate the first and last day when a license is valid. E.g.:
`:L22 :startDate "2014-01-31T00:00:00+01:00"^^xsd:dateTime .`

(a) A company is not a field.

(b) Every partner of a license is a company.

(c) Every license has one and only one operator.

(d) The company that is the operator for a license is also a partner for that license.

(e) An "active field" is defined to be a field for which there is a license that is valid *now*, i.e. 2014-05-28T15:00:00+02:00.

(f) An "operator company" is a company that is the operator of a license. (No matter whether the license is currently valid or not)

Answer:

(a) $\text{Company} \sqsubseteq \neg\text{Field}$, or
$\text{Field} \sqsubseteq \neg\text{Company}$, or
$\text{Field} \sqcap \text{Company} \sqsubseteq \bot$.

(b) $\text{License} \sqsubseteq \forall\, \text{partner.Company}$

(c) $\text{License} \sqsubseteq =_1 \text{operator}.\top$

(d) $\text{operator} \sqsubseteq \text{partner}$

(e) $\text{ActiveField} \equiv \text{Field} \sqcap \exists hasLicense.(\exists\, \text{startDate}.date[<= \text{2014-05-28}] \sqcap \exists\, \text{endDate}.date[>= \text{2014-05-28}])$

(f) $\text{OperatorCompany} \equiv \text{Company} \sqcap \exists\, \text{operator}^{-1}.\text{License}$

(Continued on page 10.)

Problem 5 RDF and OWL semantics (20 %)

(a) Consider the following axioms.

$$Person \sqsubseteq \exists father.Man \tag{1}$$
$$Person \sqsubseteq \exists mother.Woman \tag{2}$$
$$father \sqsubseteq parent \tag{3}$$
$$mother \sqsubseteq parent \tag{4}$$

Together, these axioms do *not* entail the following:

$$Person \sqsubseteq \ \geq_2 parent.\top \tag{*}$$

Give a counter-model to show that the entailment does not hold. ($\top$ is the class of all objects, the same as `owl:Thing`, so $\top^\mathcal{I} = \Delta^\mathcal{I}$)

(b) Give an axiom (5), which should not mention the *parent* relationship, such that (1)–(5) together entail (*). Explain, based on the model semantics, why the entailment holds.

(c) Give an interpretation that satisfies $A \sqsubseteq (\neg A) \sqcap B$

Answer:

(a)

$$\begin{aligned} \Delta^\mathcal{I} &= \{a, b\} \\ Person^\mathcal{I} = Man^\mathcal{I} &= Woman^\mathcal{I} = \Delta^\mathcal{I}. \\ father^\mathcal{I} = mother^\mathcal{I} &= parent^\mathcal{I} = \{\langle a, b\rangle, \langle b, a\rangle\} \end{aligned}$$

Now both ‘persons,’ a and b have only one parent, namely the other domain element, so (*) is not true in the interpretation. Since both of them are both in *Man* and *Woman*, axioms (1) and (2) are true. Also (3) and (4) are clearly satisfied.

(b) We can add $Man \sqcap Woman = \bot$ as (5), to make the two classes disjoint. Now take any interpretation $\mathcal{I}$ that makes (1)–(5) true and let $x \in Person^\mathcal{I}$. Due to (1), there exists $y \in Man^\mathcal{I}$ with $\langle x, y\rangle \in father^\mathcal{I}$, and due to (3), also $\langle x, y\rangle \in parent^\mathcal{I}$. Also, due to (2), there exists $z \in Woman^\mathcal{I}$ with $\langle x, z\rangle \in mother^\mathcal{I}$, and due to (4), also $\langle x, z\rangle \in parent^\mathcal{I}$. (5) ensures that $y \neq z$. So y and z are two distinct elements of $\{u \mid \langle x, u\rangle \in parent\}$, therefore that set has ≥ 2 elements, and $x \in (\geq_2 parent.\top)^\mathcal{I}$. Since this holds for all $x \in Person^\mathcal{I}$, we have shown that (*) holds.

(c) This is satisfied exactly if $A^\mathcal{I}$ is empty.