CO7216 Examination — Solutions

This copy generated 19th May 2017.

Title of paper	CO7216 — Semantic Web
Version	1
Candidates	All candidates
Department	Computer Science
Examination Session	MIDSUMMER EXAMINATIONS 2016
Time allowed	Two Hours
Instructions	Please do not remove this paper from the examination hall. Any number of questions may be attempted, but only the best two answers will be taken into account. Full marks may be obtained for answers to two questions. All questions carry equal weight. (50 marks per question). The exam is worth 60% of the total module mark. All relevant namespaces have been provided in the helper sheet.
Calculators	No
Books/statutes	No
Own Books/statutes/notes	No
Additional Stationery	No
Number of questions	3

Vehicle Scenario

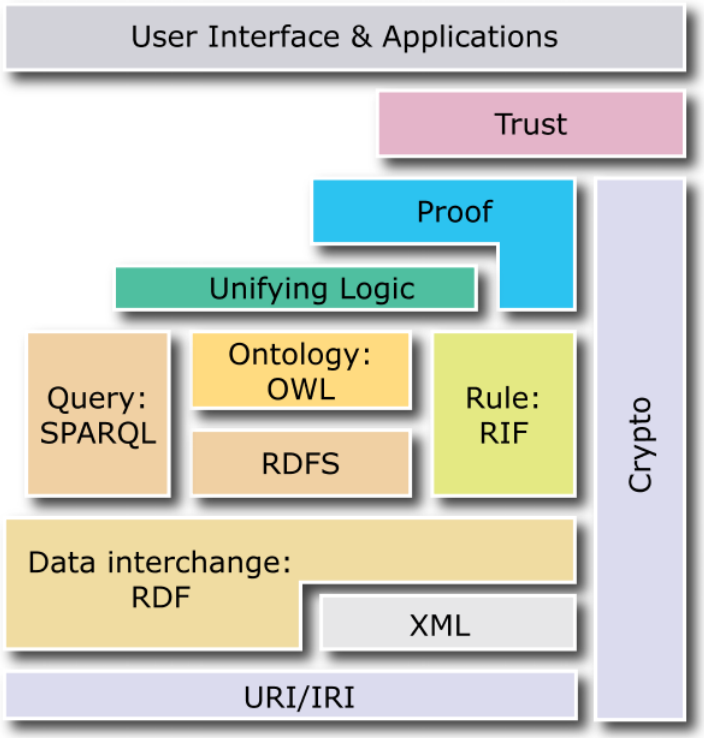
A vehicle is a mobile machine that transports people or cargo. Vehicles are either powered or unpowered. Bicycles, motor vehicles and railed vehicles are three types of vehicle. Bicycles are human-powered vehicles having two wheels attached to a frame. Motor vehicles are self-propelled road vehicles, having at least one engine, and two or more wheels, and do not operate on rails. Railed vehicles run on a prepared flat surface, and are directionally guided by the tracks on which they run. Motor vehicles can be further classified as passenger cars, buses, trucks and campervans.

A driver is a person who drives a motor or railed vehicle. In the UK, the minimum age to hold a full driving licence is 17. A passenger is a person who travels in a vehicle but not as a driver. A passenger car is a motor vehicle used for the carriage of passengers and comprising not more than eight seats in addition to a driver's seat. A bus is a motor vehicle having more than eight passenger seats. Campervans are motor vehicles that provide both transport and sleeping accommodation. Trucks are motor vehicles used for carrying goods or materials. All vehicles registered in the UK have unique registration numbers.

Question 1.

Answer to question 1

(a) The correct answer is:



[Question type: bookwork]- Lecture notes

(b) i. A possible solution:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <http://www.cs.le.ac.uk/rdf#> .

:Kate a :Employee ;
            :worksFor :NEXT .
:Alice a :Employee ;
:worksFor :HSBC .

:John a :Employee ;
:worksFor :IBM , :JP_Morgan .

:Bob a :Employee ;
:worksFor :HSBC .

:NEXT a :Retail_Comnpany ;
       :hasCompanyName "NEXT"^^xsd:string ;
       :isLocatedIn :Leicester .

:HSBC a :Bank ;
       :hasCompanyName "HSBC"^^xsd:string ;
```

```
        :isLocatedin :Leicester , :London .

  :IBM a :IT_Company ;
        :hasCompanyName "IBM"^^xsd:string ;
        :hasTelephone "+44 0116 123 0000"^^xsd:string ;
        :isLocatedIn :Leicester .

  :JP_Morgan a :Bank ;
        :hasCompanyName "JP Morgan"^^xsd:string ;
        :isLocatedIn :London .

  :Leicester a :City .

  :London a :City .
```

Note: these are **NOT** the only correct solutions. Please give credit for any other sensible answers. Credit should also be given (1) if the properties were defined in the other direction (e.g. `hasEmployee` instead of `worksFor`) or (2) if the student represents "type" as an rdf:Property rather than a Class.

[Question type: application]

ii. A possible solution:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns="http://www.cs.le.ac.uk/rdf#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://www.cs.le.ac.uk/rdf">

<rdfs:Class rdf:ID="Employee"/>

<rdfs:Class rdf:ID="Retail_Comnpany">
       <rdfs:subClassOf>
              <rdfs:Class rdf:ID="Company"/>
       </rdfs:subClassOf>
</rdfs:Class>

<rdfs:Class rdf:ID="Bank">
       <rdfs:subClassOf rdf:resource="#Company"/>
</rdfs:Class>

<rdfs:Class rdf:ID="City"/>

<rdfs:Class rdf:ID="IT_Company">
       <rdfs:subClassOf rdf:resource="#Company"/>
</rdfs:Class>

<rdf:Property rdf:ID="isLocatedIn">
       <rdfs:domain rdf:resource="#Company"/>
       <rdfs:range rdf:resource="#City"/>
</rdf:Property>

<rdf:Property rdf:ID="hasTelephone">
```

```
       <rdfs:domain rdf:resource="#Company"/>
       <rdfs:range rdf:resource="xsd:string"/>
</rdf:Property>

<rdf:Property rdf:ID="workedFor">
       <rdfs:domain rdf:resource="#Employee"/>
       <rdfs:range rdf:resource="#Company"/>
</rdf:Property>

<rdf:Property rdf:ID="hasCompanyName">
       <rdfs:range rdf:resource="xsd:string"/>
       <rdfs:domain rdf:resource="#Company"/>
</rdf:Property>
```

[Question type: application])

iii. Note: These are **NOT** the only correct solutions.

A possible SPARQL query for iii(1):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?people
WHERE {
      ?people :worksFor ?company.
      ?company rdf:type :IT_Company.
      ?company :isLocatedIn :Leicester.
}
```

A possible SPARQL query for iii(2):

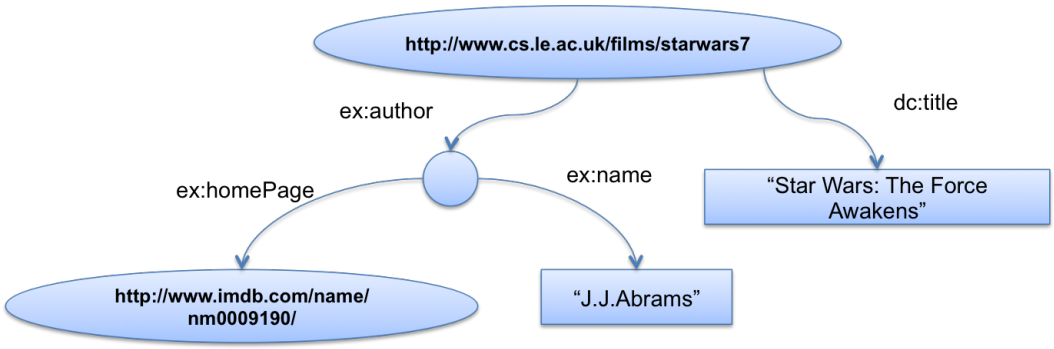
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?people ?companyName ?tel
WHERE {
    ?people  rdf:type  :Employee.
    ?people :worksFor ?company.
    ?company :hasCompanyName ?companyName.
    OPTIONAL {
    ?company :hasTelephone ?tel.
    }
}
LIMIT 4
```

A possible SPARQL query for iii(3):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  (COUNT(DISTINCT ?company) as ?count)  ?city
WHERE {
    ?company :isLocatedIn ?city.
    ?company rdf:type :Bank.
} GROUP BY ?city
```

[Question type: application])

(c) The correct answer is:



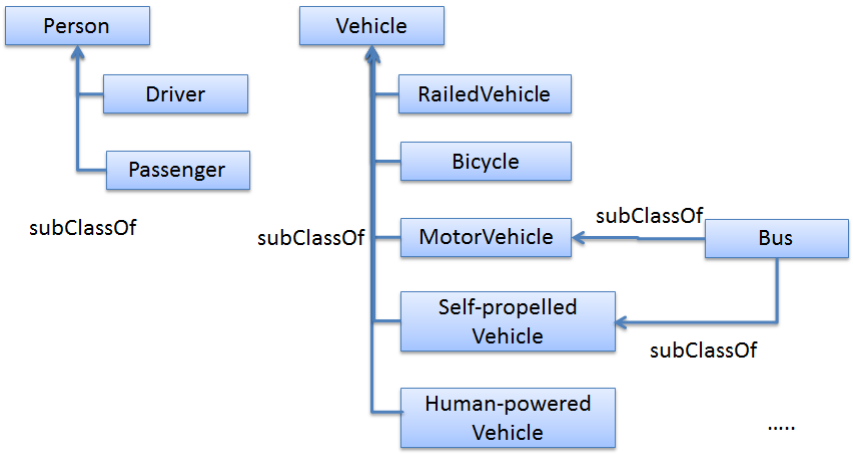
[Question type: application])

(d) The correct answers are:

- `rdfs:domain` specifies the domain of a property `P`. The class of those resources that may appear as subjects in a triple with predicate `P`.
- `rdfs:range`, specifies the range of a property `P`. The class of those resources that may appear as values (of objects) in a triple with predicate `P`.
- `rdfs:Resource`, the class of all resources.
- `rdf:Property`, the class of all properties.
- `rdf:type`, relates a resource to its classes.

(e) A possible solution:

```
Concepts: Vehicle, Bicycle, MotorVehicle, Wheel, PassengerCar,
          Truck, RailedVehicle, Driver, Passenger ...
Properties: drives (Drive/MoterVehicle), hasWheel(Vehcile/Wheel),
            runsOn (Vehcile/Surface), hasParts(Vehicle/Wheel) ....
```



[Question type: application]

Note: These are **NOT** the only correct solutions.

Question 2.

Answer to question 2

(a) A possible answer:

OWL 2 defines three new profiles or sub-languages that offer important advantages depending on your application scenario: OWL 2 EL, OWL 2 QL and OWL 2 RL. The purpose of an OWL 2 profile is to provide a trimmed down version of OWL 2 that trades expressive power for efficiency of reasoning.

Choice of profiles:

- OWL 2 EL: used for ontologies containing a large numbers of properties and/or classes
- OWL 2 QL: used for ontology-based applications that have large volumes of instance data, and where query answering is the most important reasoning task.
- OWL 2 RL: used for applications that require scalable reasoning without sacrificing too much expressive power.

[Question type: bookwork]

(b) Please refer to the lecture notes (OWL - Part 1). OWL offers some features that are not available in RDFS. For example:

Local scope of properties

Special characteristics of properties

Disjointness of classes/properties

Combinations of classes

Cardinality restrictions

...

Note: full marks should be given if the student listed at least four new features.

[Question type: bookwork/comprehension]

(c) The correct answer is:

(Bookwork, please refer to the lecture notes - OWL Part 1)

Considering the vehicle scenario. Property `hasRegistrationPlate` is a `FunctionalProperty` and an `InversedFunctionalProperty` because every vehicle must have an unique registration number and the registration can be used. Property `drives` is the inverse of `drivenBy`.

These are **NOT** the only correct solutions. Please give credit for any sensible answers.

[Question type: bookwork/comprehension]

(d) The correct answer is (i) (ii) (iv) and (vi).

(iii) is wrong because `hasFather` property is asymmetric but not transitive (e.g. if Person A `hasFather` B, B `hasFather` C then A `hasFather` C is not true).

(v) `owl:complementOf` is a set operator, should use `owl:differentFrom` to explicitly define that two individuals (instances) are different.

[Question type: comprehension]

(e) A possible answer:

```
<!-- http://www.cs.le.ac.uk/co7216/exam/#RailedVehicle -->

<owl:Class rdf:about="#RailedVehicle">
 <owl:equivalentClass>
   <owl:Class>
    <owl:intersectionOf rdf:parseType="Collection">
      <rdf:Description rdf:about="#Vehicle"/>
      <owl:Restriction>
       <owl:onProperty rdf:resource="#runOn"/>
       <owl:allValuesFrom rdf:resource="#Track"/>
      </owl:Restriction>
    </owl:intersectionOf>
   </owl:Class>
  </owl:equivalentClass>
  <rdfs:subClassOf rdf:resource="#Vehicle"/>
</owl:Class>
```

Note: the XML serialization of the OWL may vary as axioms can be expressed differently in RDF/XML. Partial credits should be given if the student uses `Restriction`, `onProperty`, `allValuesFrom`, `intersectionOf`, `subClassOf` and `equivalentClass` correctly.

[Question type: application]

(f) Note: these are **NOT** the only correct solutions. Please give credit for any sensible answers. The model solution here is written in OWL Manchester Syntax, please give credit to answers written in RDF/XML or OWL Functional syntax.

- A possible solution:

```
Class: MotorVehicle

EquivalentTo:
Vehicle
   and (poweredBy some Engine)
   and (hasWheel min 2 Wheel)

SubClassOf:
    Vehicle
```

- A possible solution:

```
Class: Vehicle

EquivalentTo:
  PowerVehicle
    or UnpoweredVehicle
```

- A possible solution:

```
Class: Bicycle

EquivalentTo:
  Vehicle
    and (poweredBy only Human)
    and (hasWheel min 2 Wheel)
```

- A possible solution:

```
Class: Bicycle

EquivalentTo:
    Vehicle
       and (poweredBy only Human)

SubClassOf:
    Vehicle
```

[Question type: application]

(g) Possible answers:

- Transitive property:
  Vehicle `isFasterThan` Vehicle;
  Vehicle `hasSameNumberOfWheelsAs` Vehicle
  ...
- Symmetric property:
  Passenger `travelingWith` Passenger;
  Vehicle `collidedWith` Vehicle
  ...
- Asymmetric property:
  Bus `hasMorePassengerSeatsThan` Bus;
  Driver `isMoreExperiencedThan` Driver
  ...
- Disjoint properties:
  MotorVehicle `hasDriver` Person
  MotorVehicle `hasPassanger` Person;
  `allowedToDrive` and `notAllowedToDrive`
  ..
- Reflexive properties:
  MotorVehicle `hasSameEngineAs` MotorVehicle
  ...
- Irrflexive properties:
  Vehicle `crashesInto` Vehicle
  ...

Note: These are **NOT** the only correct solutions. Please give credit for any sensible answers and please also consider partial credits for properties that are slightly off-topic.

Question 3.

Answer to question 3

(a) A possible answer:

```
<html>
   <head>Top Gear</head>
      <body prefix="foaf: http://xmlns.com/foaf/0.1/
                    dc: http://purl.org/dc/elements/1.1/
                    po:http://purl.org/ontology/po/">
         <div>
         <p>
          <span property="dc:title" typeof="po:Programme">Top Gear</span>
           is a <span typeof="po:BroadcasterOrganisation">BBC</span> television series
           about motor vehicles, primarily
           <span property="foaf:primaryTopic">cars</span>, and
           the most widely watched factual television programme in the world.
           The programme has received acclaim for its visual style and
           presentation  as well as criticism for its content. It is undergoing
           a major reconstruction since former presenter
           <span property="foaf:name" typeof="foaf:Person">Jeremy Clarkson</span>
          left in <span property="dc:year">2015</span>.
         </p>
         </div>
         </body>
</html>
```

...
Top Gear - [2.5 marks]
BBC [1.5 mark]
cars [1.5 mark]
Jeremy Clarkson [2.5 marks]
2015 [1 mark]
...

Note: other HTML tags such as `<div>` are also acceptable.

[Question type: application]

(b) Within the context of linked data, 303 URIs and Hash URIs are two strategies for dereferencing URIs.

303 URIs

Pros: very flexible, the redirection target can be configured separately for each resource.

Cons: multiple HTTP requests can cause latency.

- Hash URIs

  Pros: reduces access latency by reducing the number of necessary HTTP round-trips.

  Cons: a large amount of data being unnecessarily transmitted.

[Question type: comprehension]

(c) The correct answer is:

itemscope: creating the items

itemprop: adding a property to an item (used on one of the item's descendants).

itemtype: specifying he type for an item

itemid: associated an item with a global identifier

itemref: adding a property to items that are not descendants of the element.

Note: any four tags - [4 marks]; explanations [2 marks]

[Question type: bookwork]

(d) Note that these is **NOT** the only solution as SWRL/Jena rules can be written in different ways. Please give credit for any sensible answers. Possible answers:

(i) in SWRL:

```
Person(?a)^Person(?b)^isFriendOf(?a,?b)->isFriendOf(?b,?a)
```

or written as a Jena rule:

```
[rules1:(?a rdf:type Person),(?b rdf:type Person),(?a isFriendOf b)
 -> (?b isFriendOf ?a)]
```

(ii) in SWRL:

```
isYoungeThan(?a,?b)^isYoungeThan(?b,?c)->isYoungerThan(?a,?c)
```

or written as a Jena rule:

```
[rules2:(?a isYoungeThan ?b)(b? isYoungeThan ?c)->(?a isYoungeThan ?c)
```

(iii) in Jena rule

```
[rule3:  (?a ?p1 ?b), (?p1 rdfs:subPropertyOf ?p2) -> (?a ?p2 ?b)]
```

(iv) in Jena rule

```
[rule4:  (?s ?p ?o), (?p rdfs:domain ?c) -> (?s rdf:type ?c)]
```

Note: (i) and (ii) 2 marks each; (iii) (iv) 3 marks each. Partial credit should be given for partially correct answers

[Question type: application]

(e) A possible answer:

This is due to the Open-World Assumption (OWA). In contrast to the database, OWL is based on OWA. We cannot assume that if we don't know something then it is false. In this example, there may be other food that `person2` eats are not `Vegetable`. Reasoning in DL is monotonic âĂŤ if we know that `x` is an instance of `A`, then adding more information to the model cannot cause this to become false.

[Question type: comprehension]

(f) (1) Please refer to the lecture notes - (Linked) for the answers. Five steps for publishing linked data:

- Understand the Principles
- Understand your Data
- Choose URIs for Things in your Data
- Setup Your Infrastructure
- Link to other Data Sets.

(2) Apart from the predicates listed on page 40, please give credit for any sensible answers(e.g. foaf:knows ...) .

[Question type: bookwork]

(g) The correct answer is:

(ii),(iii)

(i) is wrong because `London` and `England` are instances not classes.

(ii) and (iii) - 2 marks for each option.

[Question type: comprehension]

(h) Please refer to the lecture notes (Ontology Engineering - p18).

TBox (Terminological component) : a set of classes and properties for concepts. Abox (Assertion component): individuals (or instances) belonging to those concepts. ABox statements are associated with instances of those classes.

[Question type: bookwork]

Namespaces

```
base="http://www.cs.le.ac.uk/co7216/exam/#"
rdfs="http://www.w3.org/2000/01/rdf-schema#"
owl="http://www.w3.org/2002/07/owl#"
xsd="http://www.w3.org/2001/XMLSchema#"
rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
foaf="http://xmlns.com/foaf/0.1/"
dc="http://purl.org/dc/elements/1.1/"
po="http://purl.org/ontology/po"
```

Default namespace:

```
http://www.cs.le.ac.uk/co7216/exam/#
```