M2R Exam – Semantic web: from XML to OWL
Social and semantic web part

Duration : 2h00
Any document allowed – no communication device allowed

January 2014

RDF Manipulation

Consider the following statements composing the RDF graph G:

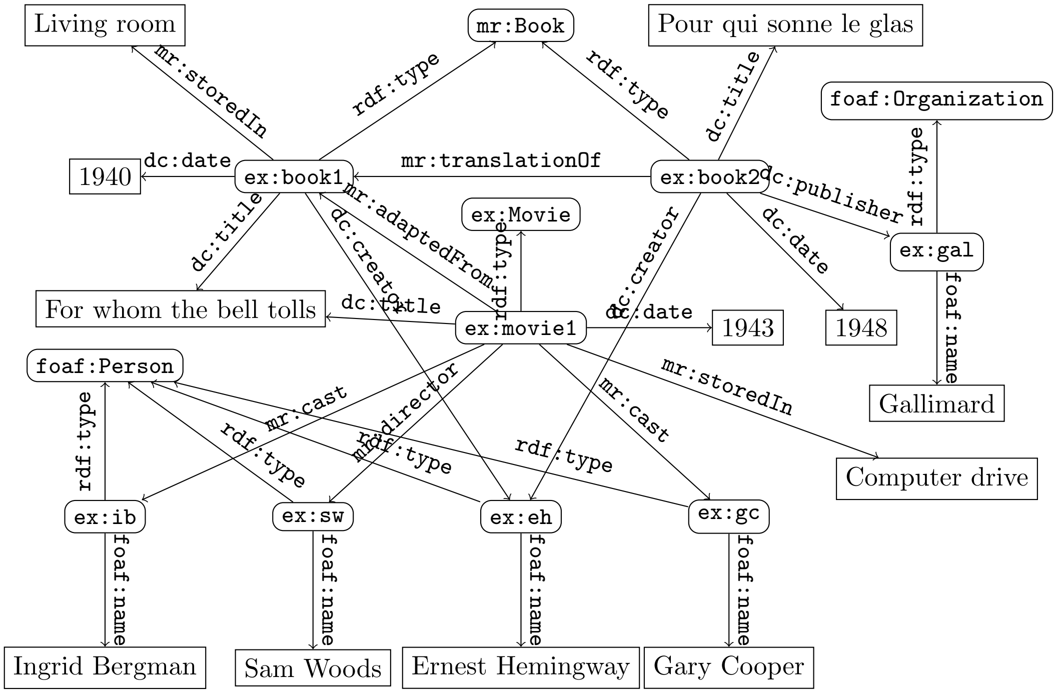
```
ex:book1 rdf:type mr:Book .
ex:book1 dc:title "For whom the bell tolls" .
ex:book1 dc:date 1940 .
ex:book1 dc:creator ex:eh .
ex:book1 mr:storedIn "Living room" .

ex:book2 rdf:type mr:Book .
ex:book2 dc:title "Pour qui sonne le glas" .
ex:book2 mr:translationOf ex:book1 .
ex:book2 dc:creator ex:eh .
ex:book2 dc:date 1948 .
ex:book2 dc:publisher ex:gal .

ex:movie1 rdf:type mr:Movie .
ex:movie1 dc:title "For whom the bell tolls" .
ex:movie1 mr:adaptedFrom ex:book1 .
ex:movie1 mr:director ex:sw .
ex:movie1 mr:cast ex:ib .
ex:movie1 mr:cast ex:gc .
ex:movie1 mr:storedIn "Computer drive" .
ex:movie1 dc:date 1943 .

ex:eh rdf:type foaf:Person .
ex:eh foaf:name "Ernest Hemingway" .

ex:sw rdf:type foaf:Person .
ex:sw foaf:name "Sam Wood .

ex:ib rdf:type foaf:Person .
ex:ib foaf:name "Ingrid Bergman" .

ex:gc rdf:type foaf:Person .
ex:gc foaf:name "Gary Cooper" .

ex:gal rdf:type foaf:Organization .
ex:gal foaf:name "Gallimard" .
```

1. Which different classes are there in graph G? Why are they classes?

   `mr:Book`, `mr:Movie`, `foaf:Person` and `foaf:Organization`.

   They are classes because they are in the range of `rdf:type`. `rdfs:Class` is the range of `rdf:type` from the RDFS Axiomatic triples. Additionnaly, they are also identified by uppercase fragments.

2. Express G as a graph in graphical form.



Consider the following as the RDF graph G' (_: are blank identifiers):

```
_:x rdf:type mr:Movie .
_:x mr:adaptedFrom _:y .
_:y dc:creator _:p .
_:p foaf:name "Ernest Hemingway" .
```

3. Paraphrase the graph G'.

   There is a movie adapted from some creation of Ernest Hemingway.

4. Is G' entailed by G? Detail why?

   Yes, $G \models G'$.

   This is because, for any model $I = \langle\rangle$ of G is such that there exist and extension ι' of ι to the blanks: _:x, _:y, _:p which assigns them `ex:movie1`, `ex:book1` and `ex:eh`. This extension satisfies: $\langle \iota'(\_:x), \iota'(mr:Movie)\rangle \in I_{EXT}(\iota'(\texttt{rdf:type}))$, $\langle \iota'(\_:x), \iota'(\_:y)\rangle \in I_{EXT}(\iota'(\texttt{mr:adaptedFrom}))$, $\langle \iota'(\_:y), \iota'(\_:p)\rangle \in I_{EXT}(\iota'(\texttt{dc:creator}))$, and $\langle \iota'(\_:p), \iota'(''ErnestHemingway'')\rangle \in I_{EXT}(\iota'(\texttt{foaf:name}))$, because I satisfies: $\langle \iota(\texttt{ex:movie1}), \iota(mr:Movie)\rangle \in I_{EXT}(\iota(\texttt{rdf:type}))$, $\langle \iota(\texttt{ex:movie1}), \iota(\texttt{ex:book1})\rangle \in I_{EXT}(\iota(\texttt{mr:adaptedFrom}))$, $\langle \iota(\texttt{ex:book1}), \iota(_{:}p)\rangle \in I_{EXT}(\iota(\texttt{dc:creator}))$, and $\langle \iota(\texttt{ex:eh}), \iota(''ErnestHemingway'')\rangle \in I_{EXT}(\iota(\texttt{foaf:name}))$,

FRBR in RDFS

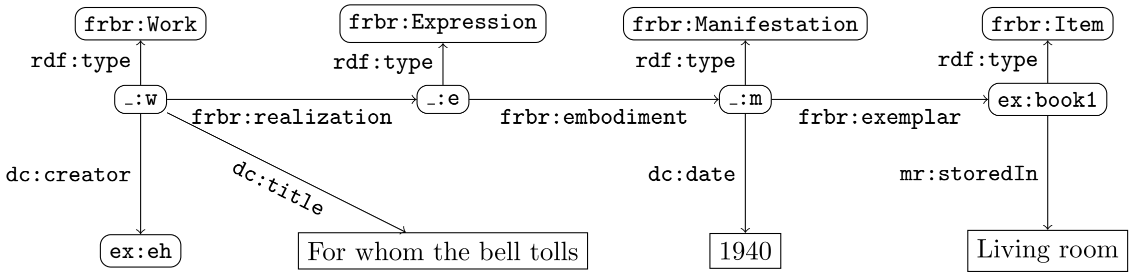
FRBR is now a well established vocabulary in libraries. FRBR distinguishes between:

- a *work* which is an abstract idea of what a work is;
- an *expression* which is a realization of this work in a particular form (poetry, music, painting);
- a *manifestation* which is a distinct embodiment of the expression (an edition of a text, a release of a movie; the reproduction of a painting);
- an *item* which is an often physical exemplar of a manifestation (an exemplar of a book; a copy of an MP3 file).

Consider a fragment S of its RDF Schema manifestation:

```
frbr:Work rdf:type rdfs:Class .
frbr:Expression rdf:type rdfs:Class .
frbr:Manifestation rdf:type rdfs:Class .
frbr:Item rdf:type rdfs:Class .
frbr:Performance rdfs:subClassOf frbr:Expression .

frbr:adaptation rdfs:domain frbr:Work .
frbr:adaptation rdfs:range frbr:Work .
frbr:realization rdfs:domain frbr:Work .
frbr:realization rdfs:range frbr:Expression .
frbr:translation rdfs:domain frbr:Expression .
frbr:translation rdfs:range frbr:Expression .
frbr:embodiment rdfs:domain frbr:Expression .
frbr:embodiment rdfs:range frbr:Manifestation .
frbr:exemplar rdfs:domain frbr:Manifestation .
frbr:exemplar rdfs:range frbr:Item .
```

For instance, the first 5 statements of G in FRBR would correspond to the following graph:



5. How would you extend the above S with the vocabulary identified by the prefix `mr` in graph G? I.e., give triples using the RDFS vocabulary (`rdfs:subClassOf`, `rdfs:subPropertyOf`, `rdfs:domain`, `rdfs:range`) to extend FRBR with the corresponding entities.

```
mr:Book rdfs:subClassOf frbr:Manifestation .
mr:Movie rdfs:subClassOf frbr:Manifestation .
mr:storedIn rdfs:domain frbr:Item .
mr:storedIn rdfs:range xsd:string .
mr:translationOf rdfs:subPropertyOf frbr:translation .
mr:translationOf rdfs:domain frbr:Expression .
mr:translationOf rdfs:range frbr:Expression .
mr:adaptedFrom rdfs:subPropertyOf frbr:adaptation .
mr:adaptedFrom rdfs:domain frbr:Work .
mr:adaptedFrom rdfs:range frbr:Work .
```

Refactoring graphs with SPARQL CONSTRUCT

6. Both movies and books may be considered as work expressions. Write a SPARQL query able to return all such items in graph *G* with their title and, if possible, the place they are stored in.

```
SELECT ?item ?title ?place
PREFIX ...
FROM G
WHERE {
  ?item dc:title ?title .
  { ?item rdf:type mr:Movie } UNION { ?item rdf:type mr:Book }
} OPTIONAL { ?item mr:storedIn ?place }
```

7. Consider that, instead of extending the schema, one would prefer to refactor the graph *G* so that it corresponds to the schema *S*. Create a SPARQL CONSTRUCT query able to extract the data from *G* and generate a graph complying to *S* (check on the example above).

```
CONSTRUCT {
  _:w rdf:type frbr:Work .
  _:w dc:creator ?creator .
  _:w dc:title ?title .
  _:w frbr:realization _:e .
  _:e rdf:type frbr:Expression .
  _:e frbr:embodiement _:m .
  _:m rdf;type frbr:Manifestation .
  _:m dc:date ?date .
  _:m frbr:exemplar ?item .
  ?item rdf:type frbr:Item .
  ?item mr:storedIn ?place .
}
PREFIX ...
FROM G
WHERE {
  ?item rdf:type mr:Book .
  ?item dc:title ?title .
  ?item dc:date ?date .
  ?item dc:creator ?creator .
  ?item mr:storedIn ?place .
}
```

8. Considering that you have a SPARQL engine able to answer queries taking into account RDFS semantics, write the same query as in Question 6 using the schema *S* and the answer to Question 5.

```
SELECT ?item ?title ?place
PREFIX ...
FROM G
WHERE {
  ?y dc:title ?title .
  ?y frbr:realization ?x .
  ?x frbr:embodiement ?e .
  ?e frbr:exemplar ?item .
} OPTIONAL { ?item mr:storedIn ?place }
```