# 1 RDF and R2RML

## 1.1 RDF, triples and prefixes

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix yago: <http://yago-knowledge.org/resource/> .
@prefix foaf: <http://xmlns.com/foaf/> .

dbo:TelevisionShow  rdfs:subClassOf  dbo:Work .

dbr:How_I_Met_Your_Mother  rdf:type  dbo:TelevisionShow .

dbr:How_I_Met_Your_Mother  rdfs:label  "How I Met Your Mother"^^xsd:string .

dbr:How_I_Met_Your_Mother  owl:sameAs  yago:How_I_Met_Your_Mother .

dbr:How_I_Met_Your_Mother  dbo:starring  yago:Alyson_Hannigan .

dbr:How_I_Met_Your_Mother foaf:name  "How I Met Your Mother"^^xsd:string .
```

Missing prefix: -1

Wrong triple format: -1

Max marks: 4

Min Marks: 0

## 1.2 Triples for R2RML mapping

(Prefixes not required but welcome)

```
dbr:How_I_Met_Your_Mother  rdf:type  dbo:TelevisionShow .

dbr:How_I_Met_Your_Mother  dbo:numberOfEpisodes  ‘‘208’’^^xsd:integer .

dbr:How_I_Met_Your_Mother  dbo:numberOfSeasons  ‘‘9’’^^xsd:integer .
```

```
dbr:How_I_Met_Your_Mother  dbo:/genre  http://dbpedia.org/resource/Romantic_Comedy .

dbr:Modern_Family  rdf:type  dbo:TelevisionShow .

dbr:Modern_Family  dbo:numberOfEpisodes  ‘‘159’’^^xsd:integer .

dbr:Modern_Family  dbo:numberOfSeasons  ‘‘7’’^^xsd:integer .

dbr:Modern_Family  dbo:/genre  http://dbpedia.org/resource/Sitcom .
```

Missing/Wrong triple format: -1

Max marks: 6

Min Marks: 0

## 1.3 R2RML mapping from triples

```
<TriplesMap1> a rr:TriplesMap;
rr:logicalTable  [rr:SQLQuery  "Select * from PLAYED-ROLE" ];
rr:subjectMap
[
  rr:template  "http://dbpedia.org/resource/{Actor}";
] ;
rr:predicateObjectMap
[
  rr:predicate  <http://dbpedia.org/ontology/playRole>;
  rr:objectMap  [ rr:template   "http://dbpedia.org/resource/{Role}" ]
] .
```

Syntax errors: -1

Wrong query in logical table: -1

Wrong subject map template: -2

Wrong predicate: -1

Wrong object template: -2

Max marks: 6

Min Marks: 0

### 1.4 Blank Nodes

(Automatically corrected I hope)

Max marks: 4

Min Marks: it can be negative, not sure if it works in the system.

## 2 SPARQL

Can be tested with: https://dbpedia.org/sparql

### 2.1 TV shows in 2016

```
SELECT DISTINCT ?show
WHERE{
  [] a dbo:TelevisionEpisode ;
  dbo:releaseDate ?date ;
  dbo:series ?series .
  ?series dbp:showName ?show
  FILTER (?date >= "2016-01-01"^^xsd:date && ?date <= "2016-12-31"^^xsd:date)
}
```

Max marks: 4

Min Marks: 0

Missing DISTINCT: -1

Missing/Wrong FILTER: -1

Missing/Wrong SELECT: -1

Missing/Wrong BGP: -1

Wrong query but good attempt: -3

### 2.2 Shows with many guests

```
SELECT ?title, count(?guest) as ?num_guests {
  [] a dbo:TelevisionEpisode ;
  dbp:title ?title ;
```

```
    dbo:guest ?guest .
  }
  GROUP BY ?title
  HAVING (count(?guest)>10)
```

Max marks: 4

Min Marks: 0

Missing COUNT: -1

Missing GROUP BY: -1

Missing HAVING: -1

Missing/Wrong SELECT: -1

Missing/Wrong BGP: -1

Wrong query but good attempt: -3

## 2.3 Ongoing TV shows

```
  CONSTRUCT {
    ?series rdf:type dbo:OngoingTVShow
  }
  WHERE {
    ?series a dbo:TelevisionShow .
    FILTER NOT EXISTS {?series dbo:completionDate ?date}
  }
```

Max marks: 4

Min Marks: 0

Missing CONSTRUCT: -1

Missing/Wrong constructed triples: -1

Missing FILTER NOT EXISTS: -1

Missing/Wrong BGP: -1

Wrong query but good attempt: -3

## 2.4 long lasting shows

```
SELECT DISTINCT ?series
WHERE {
  {
    ?series a dbo:TelevisionShow .
    ?series dbo:numberOfEpisodes ?numEpisodes .
    FILTER (?numEpisodes>200)
  }
  UNION
  {
    ?series a dbo:TelevisionShow .
    ?series dbo:releaseDate ?rdate .
    ?series dbo:completionDate ?cdate .
    BIND (YEAR(?cdate)-YEAR(?rdate) AS ?duration)
    FILTER (?duration>15)
  }
}
```

Max marks: 4

Min Marks: 0

Missing DISTINCT: -1

Missing/Wrong FILTER: -1

Missing UNION: -2

Missing/Wrong BGP: -1

Use of BIND (optional): +1

Wrong query but good attempt: -3

(*) Other solution: check release date of each episode and keep first and last (using ascending/descending order)

## 2.5 SPARQL Entailment regimes

Simple entailment: empty

RDF entailment: empty

RDFS schema: `dbr:Alyson_Hannigan` and `dbr:Joe_Manganiello`

OWL entailment: `dbr:Alyson_Hannigan` and `dbr:Joe_Manganiello`

(*) It is missing "`dbr:Josh_Radnor rdf:type Person`" so we cannot say anything about "`dbr:Josh_Radnor`" and "`freebase:Josh Radnor`"

Max marks: 4

Min Marks: 0

Wrong results for entailment regime: -1

# 3 RDFS Inference

Triples:

```
(1) :Account rdfs:subClassOf :BankService .
(2) :SavingsAccount rdfs:subClassOf :Account .
(3) :DebitAccount rdfs:subClassOf :Account .
(4) :hasAccount rdfs:domain :Customer .
(5) :hasAccount rdfs:range :Account .
(6) :hasAccount rdfs:subPropertyOf :hasService .
(7) :hasSavingsAccount rdfs:subPropertyOf :hasAccount .
(8) :hasDebitAccount rdfs:subPropertyOf :hasAccount .

(9) :sa rdf:type :SavingsAccount .
(10) :sandra :hasSavingsAccount :sa .
(11) :peter :hasAccount :ba .
(12) _:x :hasService :service .
```

**Notes on correction:**

| | |
|---|---|
| Similar, but wrong rule: | −0.5 points |
| Completely wrong or missing rule: | −1 point |
| Missing premises in rule application: | −1 point |
| Missing statement on entailment: | −0.5 points |

## 3.1 Type of `:sa`

```
(a1) :sa rdf:type :Account . (2, 9, rdfs9)
(a2) :sa rdf:type :BankService . (a1, 1, rdfs9)
```

### 3.2 Type of `:peter`

`(b1) :peter rdf:type :Customer (4, 11, rdfs2)`

### 3.3 `_:y` has service

This is not possible to derive, since nothing has itself as service. We could use simple entailment to derive `_:y :hasService _:z` from (12). but we cannot derive that `_:y` and `_:z` denotes the same resource.

### 3.4 Entailed but not derivable

E.g. `:hasAccount rdfs:range :BankService` is entailed, since the range of `:hasAccount` is a subclass of `:BankService`, but it is not derivable, since there are no RDFS entailment rule deriving a range-statement.

**Notes on correction:**

| | |
|---|---|
| Correct triple: | +3 points |
| Correct explanation on non-derivability: | +1 point |
| Correct explanation on entailment: | +1 point |

### 3.5 `:sandra` has account

```
(c1) :hasPrivateLoan rdfs:subPropertyOf :hasService . (1, 3, rdfs5)
(c2) :sandra :hasService :loan . (c1, 4, rdfs7)
```

**Notes on correction:**

| | |
|---|---|
| Stating _:r cannot be used as predicate: | +1 points |
| Giving derivation with _:r as predicate: | 1 point |

## 4 Description Logic and OWL

**Notes on correction:**

| | |
|---|---|
| Confusing $\sqsubseteq$ and $\equiv$: | $-1$ point |
| Confusing $\sqsubseteq$ and $\sqsupseteq$: | $-1$ point |
| Confusing $\sqcap$ and $\sqcup$: | $-1$ point |
| Wrong or missing $\forall$ or $\exists$: | $-1$ point |
| Missing paranthesis where necessary: | $-0.5$ points |
| Missing type (e.g. $Computer$): | $-0.5$ points |
| Confusing individual and class: | $-1$ point |

## 4.1 Computer, CPU

Manchester: `Computer EquivalentTo hasPart some CPU`
OWL : $\text{Computer} \equiv \exists \text{hasPart}.\text{CPU}$

## 4.2 Parts of computer

Manchester:
```
Computer SubClassOf
hasPart only (Motherboard or connectedTo some Motherboard)
```
OWL : $\text{Computer} \sqsubseteq \forall \text{hasPart}.(\text{Motherboard} \sqcup \exists \text{connectedTo}.\text{Motherboard})$

## 4.3 Super-computer

Manchester:
```
Computer and
((hasPart min 2 CPU) or (hasPart some (CPU and hasCore min 8 Cores)))
SubClassOf SuperCompuer
```
OWL : $\text{Computer} \sqcap (\geq_2 \text{hasPart}.\text{CPU} \sqcup \exists \text{hasPart}.(\text{CPU} \sqcap \geq_8 \text{hasCore}.\text{Core})) \sqsubseteq \text{SuperComputer}$

## 4.4 myCPU

Manchester: `(CPU and inverse hasPart some SuperComputer)(myCPU)`
OWL : $(\text{CPU} \sqcap \exists \text{hasPart}^{-}.\text{SuperComputer})(\text{myCPU})$

## 4.5 Connected to itself

Manchester: `connectedTo Self SubClassOf Nothing`
OWL : $\exists \text{connectedTo}.\text{Self} \sqsubseteq \bot$

### 4.6 Not having parts

Manchester: `hasPart o hasCore SubPropertyOf hasCore`
OWL : hasPart ∘ hasCore ⊑ hasCore

## 5 RDF and OWL Semantics

### 5.1 Blank node semantics

For A to be true in I (I|=A), there needs to be a blank node valuation beta such that I,beta|=A.

For I,beta|=hasFather(_:b,_:c) to hold with the given interpretation I, it has to be the case that beta(_:b)=joffrey and beta(_:c)=jaime, since there is only that tripe in the interpretation of hasFather.

But jaime is not in the interpretation of King, so with that beta, I,beta|=King(_:c) does not hold.

Therefore, there is no beta such that I,beta make all of A true. And therefore I does not make A true.

### 5.2 OWL Concept semantics

The interpretation of a concept expression is a subset of the domain.

a) the interpreation of "marriedTo **some** King" is the set {cersei}, since cersei is the only domain element that has a marriedTo-relation to an element of the interpretation of King.

The interpretation of "King **or** marriedTo **some** King" is the union of the interpretations of King and the previous {cersei}, so {robert, cersei}

b) the interpreation of "marriedTo **only** King" is the set of domain elements that are *not* marriedTo-related to an element that is not in the interpretation of King. Since the interpretation of King is {robert}, we are looking for domain elements that either have no marriedTo-link at all, or one to robert. The interpretation is therefore cersei, jaime, joffrey. robert is not in the set, since he is married to someone who is not a king.

The interpretation of "King **or** marriedTo **only** King" is the union of the interpretations of King and the previous set, so {robert, cersei, jaime, joffrey}.

## 5.3 OWL Axiom semantics

a) King **subClassOf** marriedTo **only** Woman

The interpretation of "marriedTo **only** Woman" is {robert, jaime, joffrey} (cersei is missing, since she is married to someone who is not in the interpretation of Woman). Since the interpretation of King is {robert}, a subset of {robert, jaime, joffrey}, the subclass axiom holds in I.

It is *not* a tautology, since there are OWL DL interpretations where it does not hold. For instance, modify I such that the interpretation of marriedTo also contains the pair <robert,robert>. Then robert is married to someone who is not a woman, and is therefore no longer in the interpretation of "marriedTo **only** Woman".

b) King **subClassOf** marriedTo **only** owl:Thing

The interpretation of "marriedTo **only** owl:Thing" in *any* interpretation will be the whole domain: there is no way that a resource x can be marriedTo-related to another resource y, and that resource is *not in the interpretation of owl:Thing*. Generally "R **only** owl:Thing" is equivalent to owl:Thing.

So the interpretation of King is a subset of that of "marriedTo **only** owl:Thing", no matter what the interpretation of King is.

This axiom holds in I as well as any other interpretation. It is an OWL tautology.