CS5560 Knowledge Discovery and Management

Problem Set 7 & 8

Submission Deadline: July 28, 2017

Name: Kallakuri Sadanand
Class ID: 9

References

I. Logical knowledge representation

First Order Logic Reference: http://pages.cs.wisc.edu/~dyer/cs540/notes/fopc.html

1) **Let us define the statements as follows:**
   - **G(x): "x is a giraffe"**
   - **F(x): "x is 15 feet or higher,"**
   - **Z(x): "x is animal in this zoo"**
   - **M(x): "x belongs to me"**

   Express each of the following statements in First-Order Logic using G(x), F(x), Z(x), and M(x).

   a) Nothing, except giraffes, can be 15 feet or higher;
   b) There is no animal in this zoo that does not belong to me;
   c) I have no animals less than 15 feet high.
   d) All animals in this zoo are giraffes.

Answer:

Possible answers are:

$\forall x(\neg G(x) \rightarrow \neg F(x))$ OR $\forall x(F(x) \rightarrow G(x))$
$\neg\exists x(Z(x) \wedge \neg M(x))$ OR $\forall x(Z(x) \rightarrow M(x))$
$\forall x(M(x) \rightarrow F(x))$
$\forall x(Z(x) \rightarrow G(x))$

2) **Which of the following are semantically and syntactically correct translations of "No dog bites a child of its owner"? Justify your answer**
   a) **∀x Dog(x) ⇒¬Bites(x, Child(Owner(x)))**
   b) **¬∃x, y Dog(x) ∧Child(y, Owner(x)) ∧Bites(x, y)**
   c) **∀x Dog(x) ⇒(∀y Child(y, Owner(x)) ⇒¬Bites(x, y))**
   d) **¬∃x Dog(x) ⇒(∃y Child(y, Owner(x)) ∧Bites(x, y))**

Answers:

b) ¬∃x, y Dog(x) ∧Child(y, Owner(x)) ∧Bites(x, y)
c) ∀x Dog(x) ⇒(∀y Child(y, Owner(x)) ⇒¬Bites(x, y))

3) For each of the following queries, describe each using Description Logic
Reference: http://www.inf.ed.ac.uk/teaching/courses/kmm/PDF/L3-L4-DL.pdf

a) Define a person is Vegan

Answer:

Value restrictions are often combined with appropriate classes using intersection:
Vegan ≡ Person ∏ ∀eats.Plant
Vegan ≡ Person ∏ ∀eats.Plant ∏ ∃eats.Plant

b) Define a person is Vegetarian

Answer:

Vegetarian ≡ Person ∏ ∀eats.(Plant U Dairy)
Vegetarian ≡ Person ∏ ∀eats.Plant ∏ ∃eats.Plant ∏ ∃eats.Diary

c) Define a person is Omnivore

Answer:

Omnivore ≡ Person ∏ ∃eats.Animal ∏ ∃eats.(Plant U Dairy)

Omnivore ≡ Person ∏ ∀eats.Plant ∏ ∃eats.Plant ∏ ∃eats.Diary ∏ ∃eats.Animal

II. SPARQL

Reference: https://www.w3.org/2009/Talks/0615-qbe/

Design a SPARQL query for following queries and show an expected output.

Query #1: Multiple triple patterns: property retrieval
Find me all the people in Tim Berners-Lee's FOAF file that have names and email addresses. Return each person's URI, name, and email address.

Answer:

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/> SELECT *
WHERE {
        ?person foaf:name ?name .
        ?person foaf:mbox ?email .
}
```

Output:

<http://www.w3.org/People/karl/karl-foaf.xrdf#me>	"Karl Dubost"	<mailto:karl@w3.org>
<http://www.w3.org/People/Berners-Lee/card#amy>	"Amy van der Hiel"	<mailto:amy@w3.org>
<http://www.w3.org/People/Berners-Lee/card#edd>	"Edd Dumbill"	<mailto:edd@xmlhack.com>
<http://www.w3.org/People/Berners-Lee/card#dj>	"Dean Jackson"	<mailto:dean@w3.org>

Query #2: Multiple triple patterns: traversing a graph
Find me the homepage of anyone known by Tim Berners-Lee.

Answer:

Query:

```
PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
PREFIX card: <http://www.w3.org/People/Berners-Lee/card#> SELECT ?homepage
```

```
FROM <http://www.w3.org/People/Berners-Lee/card> WHERE {
        card:i foaf:knows ?known .
        ?known foaf:homepage ?homepage .
}
```

Output:

http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
http://xmlns.com/foaf/0.1/Person
http://dbpedia.org/class/yago/Landmark108624891
http://dbpedia.org/class/Book
http://www.w3.org/2004/02/skos/core#Concept
http://dbpedia.org/class/yago/CoastalCities
http://dbpedia.org/class/yago/AmericanAbolitionists

Query #3: Basic SPARQL filters

Find me all landlocked countries with a population greater than 15 million.

Answer:

Query:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> PREFIX type:
<http://dbpedia.org/class/yago/>
PREFIX prop: <http://dbpedia.org/property/> SELECT
?country_name ?population
WHERE {
        ?country a type:LandlockedCountries ; rdfs:label ?country_name ;
                        prop:populationEstimate ?population .
        FILTER (?population > 15000000) .
}
```

Output:

country_name	population
Afghanistan	31889923
Afganistán	31889923
Afghanistan	31889923
Afganistan	31889923
Afghanistan	31889923

Afghanistan	31889923

Query #4: Finding artists' info

Find all Jamendo artists along with their image, home page, and the location they're near, if any.

Answer:

Query:

```
PREFIX mo: <http://purl.org/ontology/mo/> PREFIX foaf:
<http://xmlns.com/foaf/0.1/> SELECT ?name ?img ?hp ?loc
WHERE {
   ?a a mo:MusicArtist ;
        foaf:name ?name .
   OPTIONAL { ?a foaf:img ?img }
   OPTIONAL { ?a foaf:homepage ?hp } OPTIONAL {
   ?a foaf:based_near ?loc }
}
```

Output:

"Cicada"^^xsd:string	http://img.jamendo.com/artists/h/hattrickman.jpg	http://www.cicada.fr.st	http://sws.geonames.org/3031359/
"Hace Soul"^^xsd:string	http://img.jamendo.com/artists/h/hace.soul.jpg	http://www.hacesoul.com	http://sws.geonames.org/2510769/
"vincentj"^^xsd:string	http://img.jamendo.com/artists/v/vincentj.jpg	http://v.joudrier.free.fr/SiteV	http://sws.geonames.org/3020781/

Query #5. Design your own query

Answer:

Query:

```
SELECT DISTINCT ?person
WHERE {
      ?person foaf:name ?name .
```

```
        GRAPH ?g1 { ?person a foaf:Person } GRAPH ?g2
        { ?person a foaf:Person } GRAPH ?g3 { ?person a
        foaf:Person }
        FILTER(?g1 != ?g2 && ?g1 != ?g3 && ?g2 != ?g3) .
}
```

Output:

http://data.semanticweb.org/person/riichiro-mizoguchi

http://data.semanticweb.org/person/philippe-cudre-mauroux

http://data.semanticweb.org/person/lyndon-j-b-nixon

http://data.semanticweb.org/person/nigel-shadbolt

http://data.semanticweb.org/person/eero-hyvoenen

III. SWRL References:

https://www.w3.org/Submission/SWRL/

https://dior.ics.muni.cz/~makub/owl/

Design SWRL rules for the following cases

Rule #1: design hasUncle property using hasParent and hasBrother properties

Answer:

A simple use of these rules would be to assert that the combination of the hasParent and hasBrother properties implies the hasUncle property. Informally, this rule could be written as:

$$\text{hasParent}(?x1,?x2) \wedge \text{hasBrother}(?x2,?x3) \Rightarrow \text{hasUncle}(?x1,?x3)$$

Rule #2: an individual X from the Person class, which has parents Y and Z such that Y has spouse Z, belongs to a new class ChildOfMarriedParents.

Answer:

We can add a SWRL rule saying that an individual X from the Person class, which has parents Y and Z such that Y has spouse Z, belongs to a new class *ChildOfMarriedParents*. Such rule is best described in the Protege syntax:

```
Person(?x), hasParent(?x, ?y), hasParent(?x, ?z), hasSpouse(?y, ?z) -> ChildOfMarriedParents(?x)
```

Rule #3: persons who have age higher than 18 are adults.

Answer:

The following rules from the listing use the core built-ins, they would be most correctly written as:

```
Person(?p), hasAge(?p, ?age), swrlb:greaterThan(?age, 18) -> Adult(?p)
```

Rule #4: Compute the person's born in year

Answer:

```
Person(?p), bornOnDate(?p, ?date), xsd:date(?date), swrlb:date(?date, ?year,
?month, ?day, ?timezone) -> bornInYear(?p, ?year)
```

Rule #5: Compute the person's age in years

Answer:

```
Person(?p), bornInYear(?p, ?year), my:thisYear(?nowyear), swrlb:subtract(?age,
?nowyear, ?year) -> hasAge(?p, ?age)
```

Rule #6: Design your own rule

Answer:

```
Person(?x), hasChild min 1 Person(?x) -> Parent(?x)
```