CS5560 Knowledge Discovery and Management

Problem Set 7 & 8

Submission Deadline: July 28, 2017
ttps://goo.gl/forms/aTXnl4oRHMdS8j1L2

Name: Shalin Patel
Class ID: 21

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;

   (Ax) G(x)^F(x)

   b) There is no animal in this zoo that does not belong to me;

   (Ax) Z(x) v M(x)

   c) I have no animals less than 15 feet high.

   (Ax) F(x)

   d) All animals in this zoo are giraffes.
   (Ax) 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
   $\neg \exists$ x Dog(x) $\Rightarrow$ ($\exists$ y Child(y, Owner(x)) $\wedge$ Bites(x, y))

```
3) For each of the following queries,describe each using Description Logic
a)Define a person is Vegan
b)Define a person is Vegetarian
c)Define a person is Omnivore
```

$$\mathbf{Vegan} \equiv \mathbf{Person} \sqcap \forall \mathbf{eats.Plant}$$

$$\mathbf{Vegetarian} \equiv \mathbf{Person} \sqcap \forall \mathbf{eats.(Plant} \sqcup \mathbf{Dairy)}$$

$$\mathbf{Omnivore} \equiv \mathbf{Person} \sqcap \exists \mathbf{eats.Animal} \sqcap \exists \mathbf{eats.(Plant} \sqcup \mathbf{Dairy)}$$

II. SPARQL

```
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.

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

```
{
  "head": {
    "vars": [ "name" ]
  },
  "results": {
    "ordered" : true,
    "distinct" : true,
    "bindings" : [
      {
        "name" : { "type": "literal", "value": "Bijan Parsia" }
      },

      },
      {
        "name" : { "type": "literal", "value": "Tim Berners-Lee" }
      }
    ]
  }
}
```

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


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 .
}
```

concept
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
http://dbpedia.org/class/yago/AssassinatedAmericanPoliticians
http://dbpedia.org/class/yago/AssassinatedUnitedStatesPresidents
http://dbpedia.org/class/yago/Duellists
http://dbpedia.org/class/yago/IllinoisLawyers
http://dbpedia.org/class/yago/IllinoisPoliticians
http://dbpedia.org/class/yago/IllinoisRepublicans
http://dbpedia.org/class/yago/PeopleFromColesCounty,Illinois
http://dbpedia.org/class/yago/PeopleFromSpringfield,Illin

Query #3: Basic SPARQL filters
Find me all landlocked countries with a population greater than 15 million.

```
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) .
}
```

country_name	population
Afghanistan	31889923
Afganistán	31889923
Afghanistan	31889923
Afganistan	31889923
Afghanistan	31889923
Afghanistan	31889923
アフガニスタン	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.

```
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 ;
     foaf:img ?img ;
     foaf:homepage ?hp ;
     foaf:based_near ?loc .
}
```

name	img	hp	loc
"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/
"vincent j"^^xsd:string	http://img.jamendo.com/artists/v/vincentj.jpg	http://v.joudrier.free.fr/SiteV	http://sws.geonames.org/3020781/
"NoU"^^xsd:string	http://img.jamendo.com/artists/n/nou.gif	http://www.noumusic.be	http://sws.geonames.org/2802361/
"Margin of Safety"^^xsd:string	http://img.jamendo.com/artists/m/mos.jpg	http://wheresthestation.blogspot.com/	http://sws.geonames.org/660013/
"Bobywan"^^xsd:string	http://img.jamendo.com/artists/b/bobywan.jpg	http://bobywan.over-blog.org	

Query #5. Design your own query

Find me people who have been involved with at least three ISWC or ESWC conference events.

```
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) .
}
```

person

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

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

```
hasParent(?x, ?y), hasParent(?x, ?z)-> hasUncle (?x, ?z)
```

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.

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

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

```
Person(?p), integer[>= 18](?age), hasAge(?p, ?age) -> adults (?p)
```

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

```
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

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

Rule #6: Design your own rule
Hasbrother Rule:

```
Person(?p) ^ hasSibling(?p, ?s) ^ Man(?s) → hasBrother(?p, ?s)
```