CS5560 Knowledge Discovery and Management

Problem Set 7 & 8

Submission Deadline: July 28, 2017

ttps://goo.gl/forms/aTXnl4oRHMdS8j1L2

Name: Rashmi
Class ID: 29

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;

   $$\forall x(\neg G(x) \rightarrow \neg F(x)) \text{ OR } \forall x(F(x) \rightarrow G(x))$$

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

   $$\neg\exists x(Z(x) \wedge \neg M(x)) \text{ OR } \forall x(Z(x) \rightarrow M(x))$$

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

   $$\forall x(M(x) \rightarrow F(x))$$

   d) All animals in this zoo are giraffes.

   $$\forall x(Z(x) \rightarrow G(x))$$

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))

(b) and (c) are ok. (a) is bad because it uses child as a function, rather than a relation. (d) says that it's not the case that every dog bites some child of its owner

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

Vegan ≡ Person □ "eats.Plant

Vegan ≡ Person ⊓ ∀eats.Plant

b) Define a person is Vegetarian

Vegetarian ≡ Person □ "eats.(Plant □ Dairy)

Vegetarian ≡ Person ⊓ ∀eats.(Plant ⊔ Dairy)

c) Define a person is Omnivore

Omnivore ≡ Person □ #eats.Animal □ #eats.(Plant □ Dairy)

Omnivore ≡ Person ⊓ ∃eats.Animal ⊓ ∃eats.(Plant ⊔ Dairy)

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.

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

```
    ?person foaf:mbox ?email .
}
```

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 .
}
```

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) .
}
```

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 .
  OPTIONAL { ?a foaf:img ?img }
  OPTIONAL { ?a foaf:homepage ?hp }
  OPTIONAL { ?a foaf:based_near ?loc }
}
```

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) .
}
```

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(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ 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.

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

```
Artist(?x) & (≤1 artistStyle)(?x) & creator(?z,?x) ⇒ (≤1 style/period)(?z)
```