CS5560 Knowledge Discovery and Management

Problem Set 7 & 8

Name: Kalyan Kilaru

Class ID: 10

References

1. **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).

1. Nothing, except giraffes, can be 15 feet or higher;
2. There is no animal in this zoo that does not belong to me;
3. I have no animals less than 15 feet high.
4. All animals in this zoo are giraffes.
5. Which of the following are semantically and syntactically correct translations of “No dog bites a child of its owner”? Justify your answer
6. ∀ x Dog(x) ⇒ ¬Bites(x, Child(Owner(x)))
7. ¬∃ x, y Dog(x) ∧ Child(y, Owner(x)) ∧ Bites(x, y)
8. ∀ x Dog(x) ⇒ (∀ y Child(y, Owner(x)) ⇒ ¬Bites(x, y))
9. ¬∃ x Dog(x) ⇒ (∃ y Child(y, Owner(x)) ∧ Bites(x, y))
10. 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>

1. Define a person is Vegan
2. Define a person is Vegetarian
3. Define a person is Omnivore
4. **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.*

# Query #2: Multiple triple patterns: traversing a graph

*Find me the homepage of anyone known by Tim Berners-Lee.*

Query #3: Basic SPARQL filters

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

# Query #4: Finding artists' info

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

Query #5. Design your own query

1. **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

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.

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

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

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

Rule #6: Design your own rule

**SOLUTION**:.

I)

1.

a) ∀ x (¬ G(x): ¬ F(x) )

b) ∀ x (¬ Z(x): ¬ M(x) )

c) ∀ x (¬ M(x): ¬ F(x) )

d) ∀ x (¬ Z(x): ¬ G(x) )

2)

1. ∀ x Dog(x) ⇒ ¬Bites(x, Child(Owner(x)))

No dog bites dogs and owner of children.

1. ¬∃ x, y Dog(x) ∧ Child(y, Owner(x)) ∧ Bites(x, y)

No dog bits owner’s children.

1. ∀ x Dog(x) ⇒ (∀ y Child(y, Owner(x)) ⇒ ¬Bites(x, y))

All dog do not bite their children of owner.

1. ¬∃ x Dog(x) ⇒ (∃ y Child(y, Owner(x)) ∧ Bites(x, y))

Dog bite the children of owner’s.

Hence the correct answers are b and c.

3)

Vegan: People who do not eat animal products

∀ eats ¬ AnimalProducts

Vegetarian: People do not eat animal.

∀ eats ¬ Animal

Omnivore: People/ animals who eats both plant and animal

∃ eats Animal

II)

Query 1: x be the active ontology URL

SELECT \*

WHERE {

? person x:name ? name.

? person x:mbox?email.

}

# Query #2: Multiple triple patterns: traversing a graph

*Find me the homepage of anyone known by Tim Berners-Lee*

PREFIX card: <http://www.w3.org/People/Berners-Lee/card#>

PREFEX foaf: <http://xmlns.com/foaf/0.1/>

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 launches in October 1968

PREFIX space: <http://purl.org/net/schemas/space/>

PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT \*

{ ?launch space:launched ?date

FILTER (

?date > "1968-10-1"^^xsd:date &&

?date < "1968-10-30"^^xsd:date

)

}

*III)*

Rule #1:

hasParent(?x1, ?x2) ^ hasBrother(?x2, ?x3) => hasUncle(?x1,?x3)

Rule #2:

Person(?x), hasParent(?x, ?y), hasParent(?x,?y), hasSpouse(?y,?z) => ChildofMarriedParents(?x)

Rule #3:

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

Rule #4:

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

Rule #5:

Person(?p), bornInYear(?p, ?year), my:thisYear(?NowYear), swrlb: subtract(?age,?NowYear,?Year) => hasAge(?p, ?age)

Rule #6:

To generate the data, range restriction is satisfied when the ?age variable has an integer value between 18 and 65 inclusive

Person(?p), integer(>=18, <=65](?age), hasAge(?p, ?age) => hasDriverAge(?, true)