CS5560 Knowledge Discovery and Management

Problem Set 7 & 8

Name: Rakesh Reddy Pallepati

Class ID: 20

**I. Logical knowledge representation**

First Order Logic Reference: [http://pages.cs.wisc.edu/~dyer/cs540/notes/fopc.html](http://pages.cs.wisc.edu/%7Edyer/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.**

**Answer:**

**Possible answers are:**

∀*x*(¬*G*(*x*)→¬*F*(*x*)) OR ∀*x*(*F*(*x*)→*G*(*x*))

¬∃*x*(*Z*(*x*)∧¬*M*(*x*)) OR ∀*x*(*Z*(*x*)→*M*(*x*))

∀*x*(*M*(*x*)→*F*(*x*))

∀*x*(*Z*(*x*)→*G*(*x*))

1. **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)))** 
   1. **¬**∃ **x, y Dog(x)** ∧ **Child(y, Owner(x))** ∧ **Bites(x, y)**
   2. ∀ **x Dog(x)** ⇒ **(**∀ **y Child(y, Owner(x))** ⇒ **¬Bites(x, y))**
   3. **¬**∃ **x Dog(x)** ⇒ **(**∃ **y Child(y, Owner(x))** ∧ **Bites(x, y))**

**Answers:**

1. ¬∃ x, y Dog(x) ∧ Child(y, Owner(x)) ∧ Bites(x, y)
2. ∀ 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>

* 1. Define a person is Vegan **Answer:**

Value restrictions are often combined with appropriate classes using intersection:

Vegan ≡ Person ∏ ∀eats.Plant

Vegan ≡ Person ∏ ∀eats.Plant ∏ Ǝeats.Plant

* 1. Define a person is Vegetarian **Answer:**

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

Vegetarian ≡ Person ∏ ∀eats.Plant ∏ Ǝeats.Plant ∏ Ǝeats.Diary

* 1. Define a person is Omnivore **Answer:**

Omnivore ≡ Person ∏ Ǝeats.Animal ∏ Ǝeats.(Plant U Dairy)

Omnivore ≡ Person ∏ ∀eats.Plant ∏ Ǝeats.Plant ∏ Ǝeats.Diary ∏ Ǝeats.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/karlfoaf.xrdf#me> | "Karl  Dubost" | <mailto:karl@w3.org> |

|  |  |
| --- | --- |
| <http://www.w3.org/People/BernersLee/card#amy> | "Amy van der <mailto:amy@w3.org>  Hiel" |
| <http://www.w3.org/People/Berners-  Lee/card#edd> | "Edd <mailto:edd@xmlhack.com>  Dumbill" |

|  |  |  |
| --- | --- | --- |
| <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"^^xs http://img.jamendo.com/artists/h http://www.cic http://sws.geonames.or d:string /hattrickman.jpg ada.fr.st g/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/

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-mizoguch](https://www.w3.org/2009/Talks/0615-qbe/?describe=http%3A%2F%2Fdata.semanticweb.org%2Fperson%2Friichiro-mizoguchi)[i](http://data.semanticweb.org/person/riichiro-mizoguchi) [http://data.semanticweb.org/person/philippe-cudre-maurou](https://www.w3.org/2009/Talks/0615-qbe/?describe=http%3A%2F%2Fdata.semanticweb.org%2Fperson%2Fphilippe-cudre-mauroux)[x](http://data.semanticweb.org/person/philippe-cudre-mauroux) [http://data.semanticweb.org/person/lyndon-j-b-nixo](https://www.w3.org/2009/Talks/0615-qbe/?describe=http%3A%2F%2Fdata.semanticweb.org%2Fperson%2Flyndon-j-b-nixon)[n](http://data.semanticweb.org/person/lyndon-j-b-nixon) [http://data.semanticweb.org/person/nigel-shadbol](https://www.w3.org/2009/Talks/0615-qbe/?describe=http%3A%2F%2Fdata.semanticweb.org%2Fperson%2Fnigel-shadbolt)[t](http://data.semanticweb.org/person/nigel-shadbolt) [http://data.semanticweb.org/person/eero-hyvoenen](https://www.w3.org/2009/Talks/0615-qbe/?describe=http%3A%2F%2Fdata.semanticweb.org%2Fperson%2Feero-hyvoenen)

**III. SWRL**

References:

<https://www.w3.org/Submission/SWRL/>[https://dior.ics.muni.cz/~makub/owl/](https://dior.ics.muni.cz/%7Emakub/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:

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.

**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)