Name: Yalamanchili Sowmya
ClassID: 30

I Knowledge Representation

Logical knowledge representation is the field of artificial Intelligence (AI) dedicated to representing information about the world in a form that a computer system can utilize to solve complex tasks such as diagnosing a medical condition or having a dialog in natural language. Knowledge representation incorporates findings from psychology about how humans solve problems.

First order Logic (FOL or FOPC) syntax:

User defines these primitives:

1) Constant symbols (ie; "individuals in the world")
2) Function symbols (mapping individuals to individuals)
3) Predicate symbols (mapping from individuals to truth values)

FOL supplies Ex: x, y these primitives

1) Variable symbols. Ex: x, y
2) Connectives not ($\neg$), and ($\wedge$) or ($\vee$), implies ($\Rightarrow$), if and only if ($\Leftrightarrow$)
3) Quantifiers: Universal ($\forall$) and Existential ($\exists$)

(1) possible translation for the given statements 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) Syntactic Analysis:

The goal of syntactic analysis is to determine whether the text string on input is a sentence in the given natural language.

Semantic Analysis:

Semantic and pragmatic analysis make up the most complex phase of language processing as they build up on results of all the mentioned disciplines.

a) $\forall x\ Dog(x) \Rightarrow \neg Bites(x, child(owner(x)))$

No dog bites dogs and owner of children.

b) $\neg \exists x, y\ Dog(x) \wedge child(y, owner(x)) \wedge Bites(x, y)$

No dog bites owners children

c) $\forall x\ Dog(x) \Rightarrow (\forall y\ child(y, owner(x)) \Rightarrow \neg Bites(x, y)$

All dog donot bite their children of owner.

d) $\neg \exists x\ Dog(x) \Rightarrow (\exists y\ child(y, owner(x)) \wedge Bites(x, y)$

Dog bite the children of owners.

Therefore, the correct translations are (b) and (c)

(3) Description Logic: Description logic allows formal concept definitions that can be reasoned about to be expressed. It is an important element of the semantic web.

a) Define a person is vegan

people who does not eat or use animal products.

$\forall$ eats $\neg$ Animal products

b) Define a person is Vegetarian

People who does not eat animal products

$\forall$ eats $\neg$ Animal

(c) Define a person is omnivore

Animal/person eats food of both plant and animal

$\exists$ eats Animal.

Name: yalamanchili sowmya
ClassID: 30

II SPARQL:

SPARQL is the query language of the semantic web. It lets us:

1) pull values from structured and semi-structed data.
2) Explore data by querying unknown relationships.
3) perform complex joins of disparate databases in a single, simple query.
4) Transform RDF data from one vocabulary to another.

Query #1: Multiple triple patterns: property retrieval

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

Expected output: person name email

<http://www.w3.org/people/Berners-Lee/ "Amy Vander Hiel" <mailto: amy@w3.org> card#amy>

<http://www.w3.org/people/Berners-Lee/card#ds> "Dean Jackson" <mailto: dean@w3.org>

<http://www.w3.org/people/Berners-Lee/card#edd> "Edd Dumbill" <mailto: edd@usefulinc.com>

Query 2: Multiple triple patterns: traversing a graph

```
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
  }
```

Expected output:

<http://purl.org/net/eric/>
<http://www.mellon.org/about-foundation/staff/program-area-staff/irafuchs>
<http://www.johnseelybrown.com/>
<http://heddley.com/edd>

Query3: Basic SPARQL filters:

```
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 {
      ∀ eats ¬ Animal products
```

b) Define a person is vegetarian

People who doesnot eat animal

∀ eats ¬ Animal

c) Define a person is omnivore.

Animal/person eats food of both plant and Animal.

∃ eats Animal

II SPARQL:

SPARQL is the query language of the semantic web. It lets us:

1) pull values from structured and semi-structured data.
2) Explore data by querying unknown relationships.
3) perform complex joins of disparate databases in a single, simple query
4) Transform RDF data from one vocabulary to another

Query #1: Multiple triple Patterns: property retrieval

```
PREFIX foaf: <http://xmlns.com/foaf/0.1>
```

```
SELECT *
WHERE {
        ? person atype: Landlocked countries;
    rdfs: label ? country_name;
     prop: populationEstimate ? population
  FILTER (? population > 15000000)
}
```

Expected output:

country_name	population
Afghanistan	31889923
Afghanistan	31889923
⋮	
Etopia	75067000
Etopia	75067000
⋮	

Query 4: finding artists info

```
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;
}
```

wrong way

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

Rightway

Expected output

name	img	hp	loc
"ciscada"^^xsd:string	http://img.jamendo.com/artists/h/hatrickman.jpg	http://www.ciscada.fr.st	http://sws.geonames.org/3031395

"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/vincent.jpg
http://v.joudrier.free.fr/site v http://sws.geonames.org/3026781

Query 5: Design your own query

Asking a question → Is the Amazon river longer than the Nile River?

```
PREFIX PROP: <http://dbpedia.org/property/>
ASK
{
<http://dbpedia.org/resource/Amazon-River>PROP: length ?amazon
<http://dbpedia.org/resource/Nile>PROP: length ?nile
FILTER(?amazon > ?nile)
}
```

Expected output:

```
<?xml version="1.0" ?>
<sparql xmlns="https://www.w3.org/2005/sparql-results#"
    xmlns:xsi="http://www.w3.org/2001/sw/DataAccess/rf1/result2.xsd">
<head></head>
<boolean>true</boolean>
</sparql>
```

III SWRL: A semantic web Rule language

combining OWL and RuleML

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

hasParent $(?x_1, ?x_2) \wedge hasBrother(?x_2, ?x_3) \Rightarrow hasUncle(?x_1, ?x_3)$

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(?x, ?z)
childofMarriedparents (?x)

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

Person (?p), hasAge (?p, ?age), swrlb: greaterthan (?age, 18) → Adult(?p)

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

person(?P), bornOnDate (?p, ?date), xsd: date (?date), swrlb: date (?date, ?year, ?month, ?date, ?timezone) → bornInyear (?p, ?year

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

person(?p), bornInYear (?p, ?year), my: thisYear (?newYear), swrlb: subtract (?age, ?newyear, ?year] → hasAge(?p, ?age)

Rule #6: design your own rule

→ design hasSonproperty using haschild and Man properties.

haschild (?x, ?y) ∧ Man (?y) ⇒ hasSon (?x, ?y).