I

Logical Knowledge representation:

Knowledge representation and reasoning (KR) 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.

First order Logic (FOL or FOPC) syntax:

User defines these primitives:

1) Constant symbols (i.e, "individuals in the world")
2) Function symbols (mapping individuals to individuals)
3) predicate symbols (mapping from individuals to truth values)

FOL supplies these primitives:

1) variable symbols. Ex: x, y
2) Connectives: not ($\sim$), and ($\wedge$), or ($\vee$), implies ($\Rightarrow$), if and only if ($\Leftrightarrow$)
3) Quantifiers: Universal (A) and Existential (E)

1) possible translations for the given statements are

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

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 above mentioned disciplines.

a) $\forall x\ Dog(x) \Rightarrow \neg Bites(x, child(owner(x)))$
No dog bites dogs and owner of children

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

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 quering 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 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/card#amy>	"Amy vander Hiel"	<mailto:amy@w3.org>

<http://www.w3.org/people/Berners-Lee/card#dj> "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 : homepage

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

Query 3 : 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 {
```

```
?country a type: LandlockedCountries ;
         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}
```

Right way

Expected output

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

Scanned by CamScanner

"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:

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="http://www.w3.org/2005/sparql-results#"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema_instance"
        xsi:schemaLocation="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 and 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(?Y, ?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 hasChildDaughter property using hasChild and Man properties.

hasChild(?X, ?Y) ∧ Man(?Y) ⟹ hasSon(?X, ?Y)