NAME :- NAGESWARARAO CLASS ID - 18

I. Knowledge represenation:

Logical knowledge represenation 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 defining 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 (~), and (∧), or (∨), implies (=>), if and only if (<=>)

③ Quantifiers : universal (∀) and existential (∃)

Scanned by CamScanner

① possible translation for the given sentence 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 Syntatic Anaysis..

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

Semantic Anaysis :-

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

a) $\forall x\ Dog(x) \Rightarrow Bity(x, child(owner(x))]$

no dog bity dog's and owner of children

b) $\nexists x, y\ Dog(x) \wedge child(y, owner(x)) \wedge Bity (x, y)]$ No dog bity owner children

c) $\forall x\ Dog(x) \Rightarrow (\forall y\ Child(y, owner(x))) \Rightarrow \neg Bity(x,y))$

All dog donot bite their children of owner.

d) $\exists t \; x \, Dog(x) \Rightarrow \exists y \, Child(y, owner(x)) \wedge Bite(x, y)$

Dog bite the children of owners.

Therefore, the correct translation 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 eats animal

II. SPARQL :-
SPARQL is the query language of 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 pattern: property retrieval..

prefix foaf: <http://xmlns.com/foaf/1.0>

select * where { ?person foaf:name ?name
?person foaf:mbox ?email.

Expected output :- person name email

< http://www.w3.org/People/Berners-Lee/ "Amy/Anaste"

<mailto: amy@w3.org> card#amy >

<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 patterns :-

PREFIX foaf :- <http://xmlns.com/foaf/>

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

select ?homepage

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

Scanned by CamScanner

where {

card : foaf : knows ? known ?, known foaf : homepage

? homepage }

expected output :-

<http : //purl.org/net-bovics/>

<http://www. mellon. org/about-fondation/staff/

program-area-staff | saturelly >

<http : // www. johnseelybrown. com/>

< http : // heddly .com/ edd >

query 3 :- Basic SPARQL filters :-

prefix DOGS :- <http : //www. w3.org/2000p. /rdf-schema#>

prefix:coc type : <http : // dppedia . org/ class yag/>

prefix prop : <http : //dppedia .org/property/>

select ? county-name ? population />

where { ?eats ? animal products

b) Define a person is vegetarian

people who doesnt eat animal

?eats ? Animal

© Define a person is omnivore.

Animal person eats food of both plant and

animal eats animal

e Query 4 : finding artist 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	optional {?a : foaf : img ? img ?} optional {?a foaf:homepage ?hp} optional { ?a foaf: based_near ?loc }
wrong way	Right way

expected output :

" Giscard " xsd: string http://img.jamendo.com/artist/h/tuicpg.

" Kace soul " xsd: string http://img. Jamendo.com/artist/

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/rig/result-xsd">
<head> </head>
<boolean true> </boolean>
</sparql>

III SWRL :- A semantic web rule language. combining OWL and RuleML

Rule 1 : design has rule property using has parent and has Brother properties

hasparent (?x1, ?x2) ∧ hasbrother (?x2, ? x3) => hasvalue (?x1, ?x3)

Rule # 2 :- an individual x from the person class, which has parent y and z such that y has spouse z, belongs to a new person(?x), hasparent (?x, ?y), hasparent (?x, ?z), hasspouse (?x, ?z) childofmarriedparent(?x)

Rule #3 :- Persons who have age higher than 18 are adult 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(?, ?year) my thisYear (?newyear), swrlb: subtract(?age, ?newyear, ?year) → hasAge(?p, ?age)

Rule #6 :- design your own rule

→ design hasSon property using haschild and man property haschild (?x, ?y) ∧ man(?y) ⇒ hasSon(?x, ?y)

Scanned by CamScanner