Artificial Intelligence

Unit 3 : Knowledge Representation

Topics: 1)Logic Agents:

2)Knowledge Based Logic

3)Logic

4)Propositional Logic

5)First order logic:

6)Representation

7)Syntax and semantics

8)Usage

9)Knowledge Engineering

10)Inference of first order logic:

11)inference

12)Unification

13)Lifting

14)Chaining

15)Resolution

Knowledge Representation: Knowledge Representation in Artificial Intelligence - Javatpoint

- Knowledge representation and reasoning (KR, KRR) is the part of Artificial intelligence which concerned with AI agents thinking and how thinking contributes to intelligent behavior of agents.
- It is responsible for representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems such as diagnosis a medical condition or communicating with humans in natural language.
- Knowledge representation is not just storing data into some database, but it also enables an intelligent machine to learn from that knowledge and experiences so that it can behave intelligently like a human.

Techniques of knowledge representation

There are mainly four ways of knowledge representation which are given as follows:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules

LOGICAL AGENTS:

Logical Agents

Agents with some representation of complex knowledge about the world/its environment & uses inference to derive new information from the knowledge combined with new i/ps.

Knowledge Base: Set of sentences in a formal lang representing facts about the world.

Knowledge Based Agents (KBA)

-> Intelligent need knowledge about world to choose good actions/decisions

-> knowledge = {sentences} in a knowledge repⁿ language (formal lang)

-> A sentence is an assertion abt world

-> A Knowledge based agent is composed of

① knowledge Base: domain specific content

② Inference Mechanism: domain independent algorithm

-> The agents must be able to

Represent states, actions, etc

Incorporate new percepts

Update Internal repⁿ of world

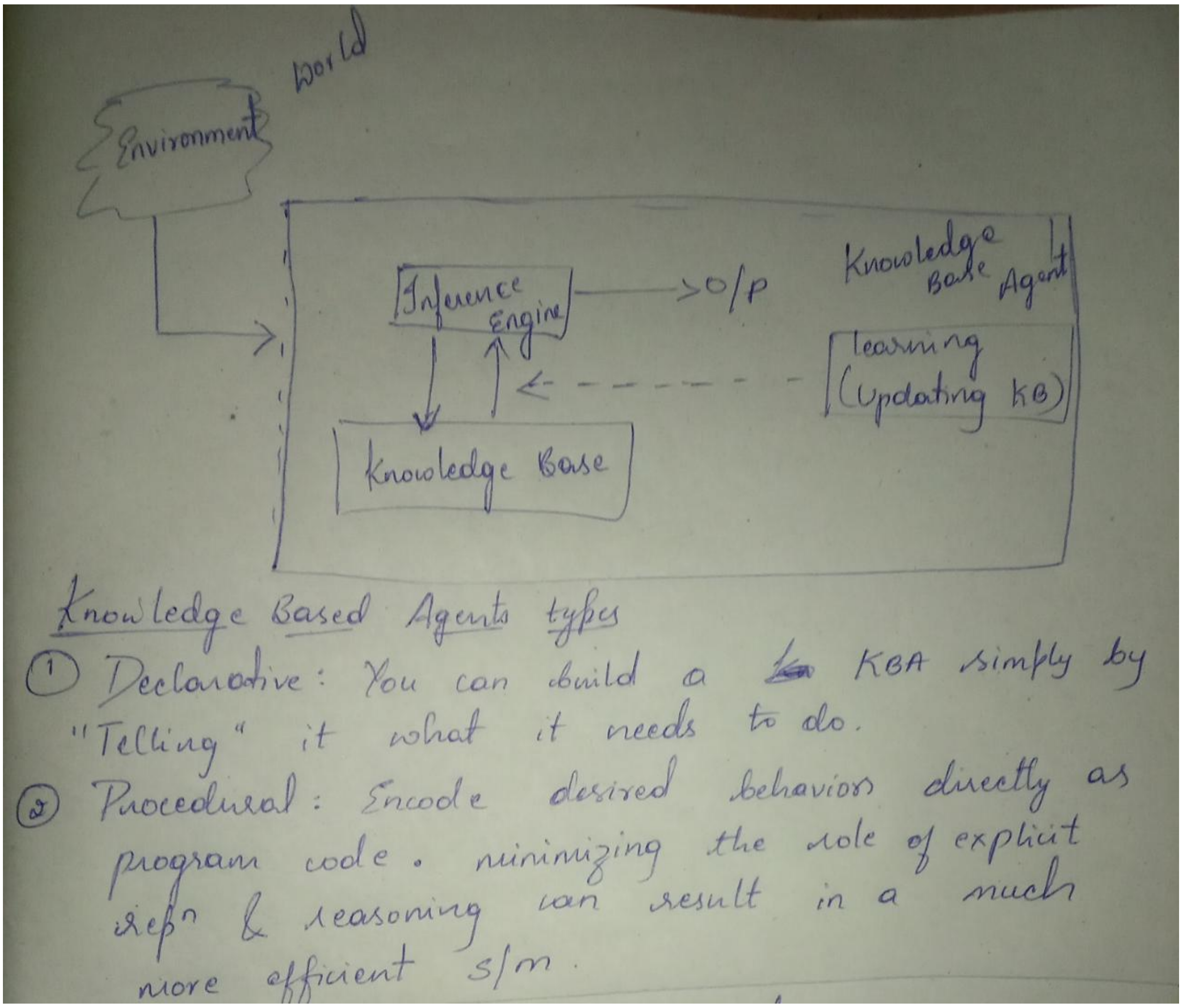
Deduce the hidden properties of world

Deduce appropriate actions

Inference Engine — Domain independent algorithm

Knowledge Base — Domain specific content

Architecture of Knowledge Based Agents



LOGICS:

- Logic can be defined as the proof or validation behind any reason provided
- While taking any decision, the agent must provide specific reasons based on which the decision was taken. And this reasoning can be done by the agent only if the agent has the capability of understanding the logic.

Types of logics in Artificial Intelligence

In artificial Intelligence, we deal with two types of logics:

1. Deductive logic
2. Inductive logic

1) Deductive logic

In deductive logic, the complete evidence is provided about the truth of the conclusion made. Here, the agent uses specific and accurate premises that lead to a specific conclusion. An example of this logic can be seen in an expert system designed to suggest medicines to the patient. The agent gives the complete proof about the medicines suggested by it, like the particular medicines are suggested to a person because the person has so and so symptoms.

2) Inductive logic

In Inductive logic, the reasoning is done through a 'bottom-up' approach. What this means is that the agent here takes specific information and then generalizes it for the sake of complete understanding. An example of this can be seen in the natural language processing by an agent in which it sums up the words according to their category, i.e. verb, noun article, etc., and then infers the meaning of that sentence.

PROPOSITIONAL LOGICS:

~~validation~~

(3) Logical Rep^n :

→ It is a language with some concrete rules that deals with propositions and has no ambiguity in rep^n.

→ It consist of syntax & ~~sy~~ sematics syntax & semantics

→ Each sentence can be translated into logic using syntax & semantics

→ Syntax : Its a well formed sentence in a language

→ Semantics: defines the truth or meaning of sentence in a would.

Logical Rep^n types

→ Propositional Logic

→ First Order predicate logic.

① Propositional Logic (PL)

-> PL is simplest logic

-> ~~Propo~~ PL is a declarative statement that is either it is true or it is false

-> Proposition logic cannot predicate it can say either true or false

=> Connectives:

Word	Symbol	Example
-> NOT	$\neg$	$\neg A$
-> and	$\wedge$	$A \wedge B$
-> OR	$\vee$	$A \vee B$
-> implies	->	$A \rightarrow B$
-> if and only if (~~Bycdir~~ (Biodirectional stmt)	<->	$A \leftrightarrow B$.

-> Truth Table

① Negation ($\neg$)

P	$\neg P$
T	F
F	T

② Conjuction ($\wedge$)

P	q	$P \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

③ disjunction ($\vee$)

P	q	$P \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

④ Implication (->)

P	q	$P \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

⑤ If & Only If (<->)

P	q	$P \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Example:

A – It is hot

B -> It is humid

C -> It is raining

Conditions:-

— If it is humid, then it is hot $B \rightarrow A$

-> If it is hot & humid then it is not raining   ~~$A \wedge B \Rightarrow \neg C$~~ $A \wedge B \rightarrow \neg C$

So, proposition is a statement of a fact.

Limitations of PL

-> We cannot represent relations like ALL, SOME or NONE with PL

a. All girls are intelligent

b. Some apples are sweet

Not a declarative statement

-> PL has limited expressive power

-> In PL, we cannot describe statements in terms of the properties or logical relationships

Example for propositional logic (Wumpus World problem)

[(64) wumpus world problem | Part-1/2| Artificial Intelligence | Lec-25 | Bhanu Priya - YouTube]

[The Wumpus world in Artificial Intelligence - Javatpoint]

Example: Propositional Logic (Wumpus World Proving)

→ The problem is, ~~to~~ statement is the agent has to find gold without falling into pit & are being eaten by wumpus & he has to come out of the cave.

Conditions

① The rooms adjacent to the wumpus are smelly/stench

② The rooms adjacent to PITs ~~to~~ has Breeze

③ There will be glitter in room if only if room has gold

~~Activators~~ Activators are: left move, right move, grab, release, shoot

Sensor: Stench, Breeze, Glitter.

B - Breeze
A - Agent
G - Gold
Ok = Safe
P = Pit
S - Stench
V = Visited
W = Wumpus

Agents 1st Step

14	24	34	44
13	23	33	43
12 OK	22	32	42
11 $\boxed{A}$ OK	21 OK	31	41

Room Safe / No Stench No Breeze

At Room [1,1], agent does not perceive any breeze or any stench, so adjacent room ok

~~As~~ Agent 2nd Step

14	24	34	44
13	23	33	43
12 OK	22 P?	32	42
11 OK	21 B $\boxed{A}$ OK	31 P?	41

→ Agent moves [2 1]

→ perceives breeze, So marks [3 1] & [2 2] as P? as they may contain pit.

→ moves back to [1 1]

Agent third step

14	24	34	44
13	23	33	43
12 S OK	22 P?	32	42
11 V A OK	21 V OK	31 P?	41

-> Now agent moves to [1 2]
-> Perceives stench so [1 3] or [2 2], [1,1] may contain wumpus.
But [1,1] was starting locⁿ of agent so it doesn't hv wumpus.
& agent ~~pe~~ didn't perceive stench at [2 1] so doesn't hv wumpus at [2 2]

—> Agent infers that [1, 3] has wumpus
—> From current state, there is no breeze so [2, 2] doesn't hv pit so agent moves to [2, 2]

Agent 4th step

14	24	34	44
13 W?	23 S B G	33	43
12 S V OK	22 V P? A	32	42
11 V OK	21 V OK	31 P?	41

Now agent perceives glitter from in [2, 3]
-> Now at [2 2] thus no stench/Breeze, So agent ~~suppose~~ suppose moves to [2 3]
-> at [2, 3] agent perceives glitter & grabs the gold

↓

14	24	34	44
13 W?	23 G S B V A	33	43
12 S V OK	22 P? V OK	32	42
11 V OK	21 V OK	31 P?	41

Found gold

Atomic Properties variable for Wompus world

-> let P_{ij} be true if there is a pit in $[i,j]$

-> let B_{ij} be true if agent perceives breeze in $[i,j]$

-> let W_{ij} be true ~~if ag~~. if there is woompus in $[ij]$

-> let S_{ij} be true if agent perceives stench in $[i,j]$

-> let V_{ij} be true if room $[ij]$ is visited

-> let G_{ij} be true if gold exist in $[ij]$

-> let OK_{ij} be true if room is safe

4	14	24 P?	34	44
3	13 W?	23 S G B	33	43
2	12	22 V P?	32	42
1	$\boxed{A}$ 11 OK	21 V B OK	31 P?	41
	1	2	3	4

Propositional Rules for Wompus world

$R_1 \rightarrow \neg S_{11} \rightarrow \neg W_{21} \wedge \neg W_{11} \wedge \neg W_{12}$

$R_2 \rightarrow \neg S_{21} \rightarrow \neg W_{11} \wedge \neg W_{21} \wedge \neg W_{31} \wedge \neg W_{22}$

$R_3 \rightarrow \neg S_{12} \rightarrow \neg W_{11} \wedge \neg W_{13} \wedge \neg W_{22} \wedge \neg W_{12}$

$R_4 \rightarrow S_{12} \rightarrow W_{13} \vee W_{22} \vee W_{12} \vee W_{11}$

Prove that Wampus is in room (1,3)

-> Apply Modus ponens with $\neg S_{11}$ & R_1

$\boxed{\neg S_{11} \rightarrow \neg W_{21} \wedge \neg W_{11} \wedge \neg W_{12}}$ $\boxed{\neg S_{11}}$

$\boxed{\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}}$

-> ~~Apply Modus~~ ponens

After AND Elimination rule to $\neg W_{11} \wedge \neg W_{12} \wedge \neg W_{21}$ we get $\neg W_{11}$, $\neg W_{12}$, & $\neg W_{21}$

-> Apply Modus Ponens with $\neg S_{21}$ & R_2

$\boxed{\neg S_{21} \rightarrow \neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}}$ $\boxed{\neg S_{21}}$

$\boxed{\neg W_{21} \wedge \neg W_{22} \wedge \neg W_{31}}$

On Applying and - Elimination rule we get
~~to~~ $\neg W_{21}$, $\neg W_{22}$ & $\neg W_{31}$

-> Apply MP with S_{12} & R_4

$\boxed{S_{12} \rightarrow W_{13} \vee W_{12} \vee W_{22} \vee W_{11}}$ $\boxed{S_{12}}$

$\boxed{W_{13} \vee W_{12} \vee W_{22} \vee W_{11}}$

~~apply and elimination Rule~~
~~we get~~ ~~W_{13}, W_{12}~~.

-> Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22} \vee W_{11}$ & $\neg W_{11}$

$\boxed{W_{13} \vee W_{12} \vee W_{22} \vee W_{11}}$ $\boxed{\neg W_{11}}$

$\boxed{(W_{13} \vee W_{12} \vee W_{22})}$

-> Apply Unit resolution on $W_{13} \vee W_{12} \vee W_{22}$ & $\neg W_{22}$

$\boxed{W_{13} \vee W_{12} \vee W_{22}}$ $\boxed{\neg W_{22}}$

$\boxed{W_{13} \vee W_{12}}$

-> Apply Unit resolution on $W_{13} \vee W_{12}$ with $\neg W_{12}$

$\boxed{W_{13} \vee W_{12}}$ $\boxed{\neg W_{12}}$

$\boxed{W_{13}}$ Proved

First Order Logic

- First-order logic is another way of knowledge representation in artificial intelligence. It is an extension to propositional logic.

- First-order logic is also known as Predicate logic or First-order predicate logic. First-order logic is a powerful language that develops information about the objects in a more easy way and can also express the relationship between those objects.

- First-order logic (like natural language) does not only assume that the world contains facts like propositional logic but also assumes the following things in the world:
  - Objects: A, B, people, numbers, colors, wars, theories, squares, pits, wumpus, ...
  - Relations: It can be unary relation such as**:** red, round, is adjacent, or n-any relation such as: the sister of, brother of, has color, comes between
  - Function**:** Father of, best friend, third inning of, end of, ...
- As a natural language, first-order logic also has two main parts:
  a. Syntax
  b. Semantics

- ## Syntax and semantics of FOL
- basic Elements of FOL:

Constant	1, 2, A, John, Mumbai, cat,....
Variables	x, y, z, a, b,....
Predicates	Brother, Father, >,....
Function	sqrt, LeftLegOf, ...
Connectives	∧, ∨, ¬, ⇒, ⇔
Equality	==
Quantifier	∀, ∃

Atomic sentences:

- Atomic sentences are the most basic sentences of first-order logic. These sentences are formed from a predicate symbol followed by a parenthesis with a sequence of terms.
- We can represent atomic sentences as Predicate (term1, term2, ..., term n).

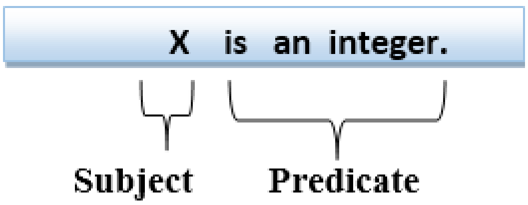
Example: Ravi and Ajay are brothers(sentence): => Brothers(Ravi, Ajay) (atomic sentence).
Chinky is a cat: => cat (Chinky).

Complex Sentences:

- Complex sentences are made by combining atomic sentences using connectives.

First-order logic statements can be divided into two parts:

- Subject: Subject is the main part of the statement.
- Predicate: A predicate can be defined as a relation, which binds two atoms together in a statement.
- Consider the statement: "x is an integer.",
- it consists of two parts, the first part x is the subject of the statement and second part "is an integer," is known as a predicate.



Quantifiers in First-order logic:

- A quantifier is a language element which generates quantification, and quantification specifies the quantity of specimen in the universe of discourse.
- These are the symbols that permit to determine or identify the range and scope of the variable in the logical expression. There are two types of quantifier:
  a. Universal Quantifier, (for all, everyone, everything)
  b. Existential quantifier, (for some, at least one).

Universal Quantifier:

Universal quantifier is a symbol of logical representation, which specifies that the statement within its range is true for everything or every instance of a particular thing.

The Universal quantifier is represented by a symbol

If x is a variable, then ∀x is read as:

- For all x
- For each x
- For every x.

Example:

All man drink coffee.

∀x man(x) → drink (x, coffee).

It will be read as: There are all x where x is a man who drink coffee.

Existential Quantifier:

Existential quantifiers are the type of quantifiers, which express that the statement within its scope is true for at least one instance of something.

It is denoted by the logical operator ∃

If x is a variable, then existential quantifier will be ∃x or ∃(x). And it will be read as:

- There exists a 'x.'
- For some 'x.'
- For at least one 'x.'

Example:

Some boys are intelligent.

∃x: boys(x) ∧ intelligent(x)

It will be read as: There are some x where x is a boy who is intelligent.

Some Examples of FOL using quantifier:

1. All birds fly.
In this question the predicate is "fly(bird)."
And since there are all birds who fly so it will be represented as follows.

$\forall x$ bird(x) $\rightarrow$fly(x).

2. Every man respects his parent.
In this question, the predicate is "respect(x, y)," where x=man, and y= parent.
Since there is every man so will use $\forall$, and it will be represented as follows:

$\forall x$ man(x) $\rightarrow$ respects (x, parent).

3. Some boys play cricket.
In this question, the predicate is **"**play(x, y)," where x= boys, and y= game. Since there are some boys so we will use **∃**, and it will be represented as:

$\exists x$ boys(x) $\rightarrow$ play(x, cricket).

4. Not all students like both Mathematics and Science.
In this question, the predicate is "like(x, y)," where x= student, and y= subject.
Since there are not all students, so we will use $\forall$ with negation**, so** following representation for this:

$\neg\forall$ (x) [student(x) $\rightarrow$ like(x, Mathematics) $\wedge$ like(x, Science)].

Knowledge Engineering:

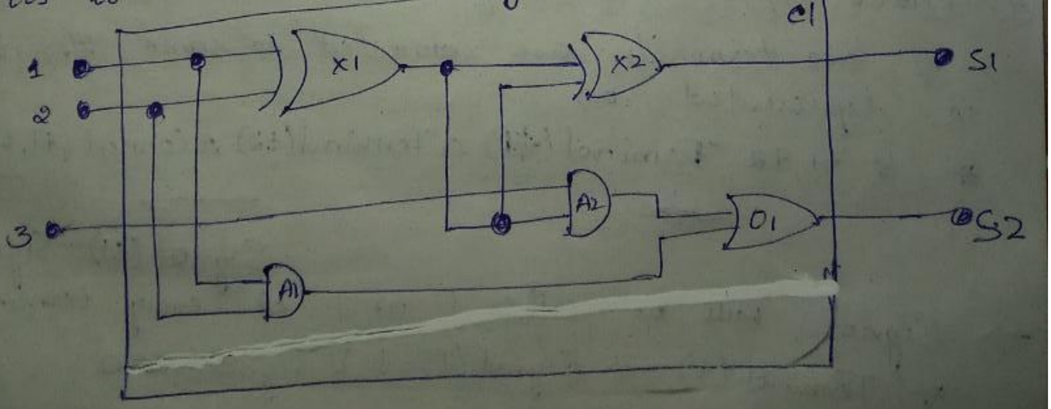
Knowledge Engineering

-> The process of constructing a knowledge-base in first-order-logic is called Knowledge Engineering
-> The person who investigates & learns about domain and generates repn of a object is called knowledge Engineer
-> Following are main steps in knowledge engineering process

~~① Identify the task ; This step is the process of identifying the task.~~
~~② Assemble the relevant knowledge :~~

① Identify the task
② Assemble relevant knowledge
③ Decide on Vocabulary
④ Encode general knowledge about domain.
⑤ Encode a description of the problem instance
⑥ Pose queries to the inference procedure & get answers
⑦ Debug the knowledge base :

Ex: let us develop a knowledge base which will us to reason abt digital C^t (1 bit full adder)



① Identify the task (circuit)

* At 1st level, examine functionality of C^t such as
  - Does the C^t add properly?
  - What will be o/p of gate A2, if all i/p s are high?
* At 2nd level, examine C^t Structure such as
  - which gate is connected to 1st i/p terminal?
  - Does the C^t have feedback loops

② Assemble the relevant knowledge

-> Assemble required for digital C^{ts} such as
  - logic C^t are made up of wires and gates
  - They are 4 types of gates used i.e AND, OR, XOR & NOT
  - All these gates have 2 ~~terminal~~ i/p & 1 o/p

③ Decide On Vocabulary

-> This step involves process of selecting functions, predicates & constants to represent C^{ts}, terminals, signals and gates

-> Gate represented as Gate(X1)

-> Terminal identified by predicate Terminal (X)

-> C^t identified by predicate Circuit(C1)

-> For gate i/p, we use funⁿ In(1, X1) & ~~for o/p OUT~~ for o/p OUT(1, X1)

-> Connectivity b/w gates represented by predicate
Connect (OUT(1, X1) , IN(1, X1))

-> If signal is on is given by predicate On(t)

④ Encode General knowledge abt the domain

-> If two terminals are connected to same i/p, it is represented as

* ∀ t1, t2 Terminal (t1) ∧ Terminal(t2) ∧ Connect (t1, t2) → Signal (t1) = Signal(t2)

-> Signal will be either 0 or 1 at every terminal

∀ Terminal (t) → Signal (t) = 1 ∨ Signal(t) = 0

→ Connect predicates are commutative

~~∀ t1, t2 Connect (t1) →~~

$\forall\ t1, t2$ Connect $(t1, t2) \rightarrow$ Connect $(t2, t1)$
and ~~so on~~ etc

(5) Encode description of pblm instance

→ We categorize ckt & gate components & write simple atomic sentences of instances of concepts which is known as ontology.

→ In the ckt there are 2 XOR, 2 AND & 1 OR gate so atomic sentences are

For XOR gate: Type (X1) = XOR, Type (X2) = XOR

For AND gate: Type (A1) = AND, Type (A2) = AND

For OR gate: Type (O1) = OR

(6) Pose Queries to the inference procedure & get answers

→ We find all possible set of values of all the terminal for the adder circuit.

$\exists\ i1, i2, i3$ Signal (In (1, C1)) = i1
$\wedge$ Signal (In (2, C1)) = i2
$\wedge$ Signal (In (3, C1)) = i3
$\wedge$ Signal (Out (1, C1)) = 0 $\wedge$ Signal (Out (2, C1)) = 1

(7) Debug Knowledge Base

→ We will debug the issues in knowledge Base in the above ckt knowledge base, we may have omitted assertions like $1 \neq 0$.

Inference in fol

Inference in First-Order Logic is used to deduce new facts or sentences from existing sentences. Before understanding the FOL inference rule, let's understand some basic terminologies used in FOL.

Substitution:

Substitution is a fundamental operation performed on terms and formulas. It occurs in all inference systems in first-order logic.

If we write **F[a/x]**, so it refers to substitute a constant "**a**" in place of variable "**x**".

Equality:

First-Order logic does not only use predicate and terms for making atomic sentences but also uses another way, which is equality in FOL.

Example: Brother (John) = Smith.

As in the above example, the object referred by the **Brother (John)** is similar to the object referred by **Smith**.

Example: ¬(x=y) which is equivalent to x ≠y.

FOL inference rules for quantifier:

following are some basic inference rules in FOL:

- **Universal Generalization**
- **Universal Instantiation**
- **Existential Instantiation**
- **Existential introduction**

1. Universal Generalization:

- Universal generalization is a valid inference rule which states that if premise P(c) is true for any arbitrary element c in the universe of discourse, then we can have a conclusion as ∀ x P(x).
- It can be represented as: $\frac{P(c)}{\forall x\, P(x)}$.

  Example: Let's represent, P(c): "**A byte contains 8 bits**", so for ∀ **x P(x)** "**All bytes contain 8 bits.**", it will also be true.

2. Universal Instantiation(elimination):

- Universal instantiation is also called as universal elimination It can be applied multiple times to add new sentences.
- we can infer any sentence P(c) by substituting a ground term c (a constant within domain x) from ∀ **x P(x) for any object in the universe of discourse**.
- It can be represented as: $\dfrac{\forall x\, P(x)}{P(c)}$.

Example:1.

IF "Every person like ice-cream"=> ∀x P(x) so we can infer that
"John likes ice-cream" => P(c)

3. Existential Instantiation:

- Existential instantiation is also called as Existential Elimination
- It can be applied only once to replace the existential sentence.
- This rule states that one can infer P(c) from the formula given in the form of ∃x P(x) for a new constant symbol c.
- It can be represented as: $\dfrac{\exists x\, P(x)}{P(c)}$

4. Existential introduction

- An existential introduction is also known as an existential generalization
- This rule states that if there is some element c in the universe of discourse which has a property P, then we can infer that there exists something in the universe which has the property P.
- It can be represented as: $\dfrac{P(c)}{\exists x P(x)}$
- **Example: Let's say that,**
  "Priyanka got good marks in English."
  "Therefore, someone got good marks in English."

Unification in fol

~~Inference~~

Unification

→ It is all abt making the expression look identical. So, for the given expression to make them look identical we need to do substitution

Eg: $P(x, F(y))$ — ①, $P(a, F(g(z)))$ ②

Unification : $[a/x, g(z)/y]$ $x = a$, $y = g(z)$

x substituted with a

y " " " g(z)

→ In abv Eg, Substitute x with a & y with g(z), it is represented as $a|x$ & $g(z)|y$

→ In both Expⁿ, 1st Expⁿ is identical to 2nd expⁿ & the substitution set will be $[a|x, g(z)|y]$

Conditions for Unification

→ Predicate symbol must be same, atoms or Expⁿ with diff predicate symbol will never be unified

→ No of arguments in both Expⁿ must be identical

→ Unification will fail, if there are 2 similar variables present in same expⁿ.

Unification Algorithm

⇒ Algorithm : Unify (L_1, L_2)

Step1: If L_1 & L_2 is a variable or constant, then :

(a) If L_1 & L_2 are identical return NIL

(b) Else if L_1 is a variable, then if L_1 occurs in L_2 then return FAIL Else return $\{(L_2/L_1)\}$

(c) Else if L_2 is a variable, then if L_2 occurs in L_1 then return FAIL, else return $\{(L_1/L_2)\}$

(d) Else return FAIL

Step 2: If the initial predicate symbol is L_1 & L_2 are not identical, then return FAIL

Step 3: Suppose if $L1$ & $L2$ have a diff no of argument then return FAIL

Step 4: SET [illegible] SUBST[illegible] TO NIL

Step 5: LOOP ⟹ { for $i \leftarrow 1$ to no of arguments of L_1

a) Call unify with the i^{th} argument of L_1 and the ith argument of L_2 putting result in S

b) ~~Supp~~ If S = FAIL then return FAIL

c) If S is not Equal to ~~FA~~ NIL then

d) Apply S to remainder of both L_1 & L_2

e) SUBST = APPEND (S_1, SUBST) }

Step 6: Return SUBST

Implementation of the Algorithm

Step 1: Initialize the Substitution set to be empty

Step 2: Recursively unify atomic sentences

a) Check for identical expression match

b) If one Exp^n is a variable v_i & other is a term t_i which does not contain variable v_i then:

→ Substitute t_i/v_i in existing substitutions

→ Add t_i/v_i to the substitution setlist

→ If both exp^n are fun^n, then fun^n name must be similar & the no of arguments must be the same in both the exp^n.

Example

Consider $P(x, g(x))$

Sol^n

i) $P(z, y)$: ~~Unify~~ Unifies with $[x/z, g(x)/y]$

$x = z \quad g(x) = y$

ii) $P(z, g(z))$: Unifies with $[x/z, z/x]$ $x = z, g(x) = g(z)$

iii) $P(\text{prime}, F(\text{prime}))$: does not Unifies

Ǝ fun^n f & g does not match

Resolution in fol

Resolution In FOL

→ Resolution is a theorem proving technique that uses proofs by contradiction

→ It is used, if there are various stmts given & need to prove a conclusion of those stmt

→ Unification is a key concepts in proofs by resolution

→ Resolution is a single inference rule which can efficiently operate on conjunctive normal form or casual form

Clause : Disjunction of literals (an atomic sentence) is called a clause

Conjuctive NF : A sentence represented as a Conjuction of clauses said to be CNF.

Steps for Resolution

* Conversion of facts into Fol
* Convert FOL stmt into CNF
* Negate the stmt which needs to prove (proof by Contradiction)
* Draw resolution graph (unification)

Example :

a. John likes all kinds of food

b. Apple & Vegetable are food

c. Anything anyone eats & not killed is food

d. ~~Ho~~ Anil eats peanuts & still alive

e. Harry eats everything that Anil eats

Prove by resolution that :

f. John likes peanuts

Steps 1 : Conversion of facts into FOL

a. $\forall x$: food(x) $\rightarrow$ likes (John, x)

b. food (Apple) $\wedge$ food (vegetables)

c. $\forall x \forall y$: eats (x, y) $\wedge$ $\neg$ Killed (x) $\rightarrow$ food (y)

d. eats (Anil, peanuts) $\wedge$ alive [Anil)

e. $\forall x$: eats (Anil, x) $\rightarrow$ eats (Harry, x)

f. $\forall x$: $\neg$ Killed (x) $\rightarrow$ alive (x)

g. $\forall x$: ~~alive~~ alive (x) $\rightarrow$ $\neg$ Killed (x) } added predicates

h. likes (John, peanuts)

Step2 : Conversion of FOL into CNF

$\because$ CNF makes easier for CNF proofs.)

i) Eliminate all implications ($\rightarrow$) & rewrite

a $\Rightarrow$ b
$\neg$(a v b)

[a $\rightarrow$ b = $\neg$(a v b)]

a. $\forall x$: $\neg$ food (x) v likes (John, x)

b. food (Apple) $\wedge$ food (vegetables)

c. $\forall x \forall y$: $\neg$ [eats (x, y) $\wedge$ $\neg$ killed (x)] v food y)

d. eats (Anil, peanuts) $\wedge$ alive (Anil)

e. $\forall x$: $\neg$ eats (Anil, x) v eats (Harry, x)

f $\forall x$: $\neg$ [$\neg$ killed (x)] v alive (x)

g. $\forall x$: $\neg$ alive (x) v ($\neg$ killed (x))

h. likes (John, peanuts)

ii) Move negation ($\neg$) inwards & rewrites

~~a. $\forall x$: $\neg$ food (x) $\wedge$ $\neg$ likes (John, x)~~

a. $\forall x$: $\neg$ food (x) v likes (John, x)

b. food (Apple) $\wedge$ food (Vegetables)

c. $\forall x \forall y$: $\neg$ eats (x, y) v Killed (x) v food(y)

d. eats (Anil, peanuts) $\wedge$ alive (Anil)

e $\forall x$: $\neg$ eats (Anil, x) v eats (Harry, x)

f $\forall x$: $\neg$ Killed (x) v alive (x)

g $\forall x$: $\neg$ alive x) v $\neg$ killed (x)

h. likes (John, Peanuts)

iii) Rename Variables

a. $\forall x \; \neg$ food(x) $\vee$ likes(John, x)

b. food (Apple) $\wedge$ food (vegetables)

c. $\forall y \forall z \; \neg$ eats (y, z) $\vee$ Killed(y) $\vee$ food(z)

d. eats (Anil, Peanuts) $\wedge$ alive (Anil)

e. $\forall w \; \neg$ eats (Anil, w) $\vee$ eats (Harry, w)

f. $\forall g \; \neg$ Killed(g) $\vee$ alive(g)

g. $\forall k \; \neg$ alive (k) $\vee$ $\neg$ killed (k)

h. likes (John, Peanuts)

iv) Eliminate Existential instantiation quantifier by Elimination

There is no Existential instantiation $\exists$ quantifier so ~~everything~~ all stmts remain same

v) Drop Universal Quantifier

~~[illegible]~~ food

a. $\neg$food(x) $\vee$ likes(John, x)

b. food (Apples)

c. food (vegetables)

d. $\neg$eats(y, z) $\vee$ killed(y) $\vee$ food(z)

e. eats (Anil, Peanuts)

f. alive (Anil)

g. $\neg$eats (Anil, w) $\vee$ eats (Harry, w)

h. killed (g) $\vee$ alive (g)

i. $\neg$ alive(k) $\vee$ $\neg$killed (k)

J. likes (John, Peanuts)

Gold

vi) Distribute conjunction $\wedge$ over disjunction $\neg$

This step will not make any change to this problem.

Step 3 : Negate the stmt to be proved

In this step we apply negation to conclusion stmt,

ie ¬likes(John, Peanuts)

Step 4 : Draw Resolution Graph

Now, in this step, we will solve the problem by resolution tree using substitution.

¬likes(John, Peanuts)   ¬food(x) V likes(John, x)

{peanuts/x} ⓐ

¬food(peanuts)   ¬eats(y, z) V killed(y) V food(z) ⓒ

{peanuts/z}

¬eats(y, peanuts) V killed(y)   eats(Anil, Peanuts)

{Anil/y} ⓔ

killed(Anil)   ¬alive(k) V ¬killed(k)

{~~Anil~~ Anil/k} ⓘ

¬alive(Anil)   alive(Anil) ⓕ.

{ } Hence proved

Hence the negation of conclusion has been proved as a complete contradiction with given set of stmts.

Resolution Example

If it is Sunny & warm day you will Enjoy

If it is rainy you will get

It is warm day

It is rainy

It is sunny

Goal : You will Enjoy.

Step 1 : Convert ~~into~~ facts into F.OL

① If it is sunny & warm day you will Enjoy

FOL Sunny ∧ warm → Enjoy

② If it is rainy you will get

rainy → wet

③ It is warm day

warm

④ It is rainy

rainy

⑤ It is sunny ⑥ Enjoy

sunny

~~⑥ you will Enjoy~~

Step 2 : Convert FOL stmt into CNF

[a → b = ¬ a ∨ b]

* ¬ (sunny ∧ warm) ∨ Enjoy

CNF ¬sunny ∨ ¬warm ∨ Enjoy

* ~~rainy → w~~et ¬~~rainy wet~~ CNF ¬rainy ∨ wet

* warm

* ~~Sunny~~ rainy

* Sunny

Step 3 : negate the stmt to be proved

* ¬ Enjoy

Step 4 : Draw resolution Graph

¬ Enjoy ¬ Sunny ∨ ¬ warm ∨ Enjoy ①

¬Sunny ∨ ¬warm warm

¬Sunny Sunny

{ } Hence proved (Contradiction

Chaining in fol

Forward chaining/reasoning

→ It is a form of reasoning which starts with atomic sentences in the knowledge base & applies inference rules in the forward direction to Extract more data until a goal is reached

Properties

→ Moves from bottom to top

→ It is a process of making a conclusion based on known facts or data by ~~selecting~~ fr starting from the initial state & reach the goal state

→ It is also called as data-driven as we reach goal using available data

→ Forward chaining used in Expert s/m

Example 1:

Rule 1: If A & C then F

2: If A & E, then G

3: If B then E

4: If G then D

Knowledge Base

A & C → F

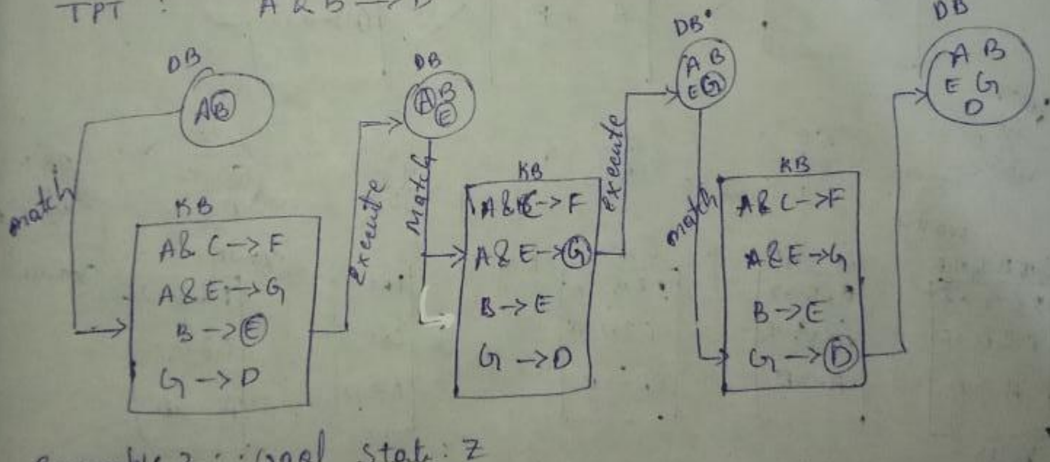
A & E → G

B → E

G → D

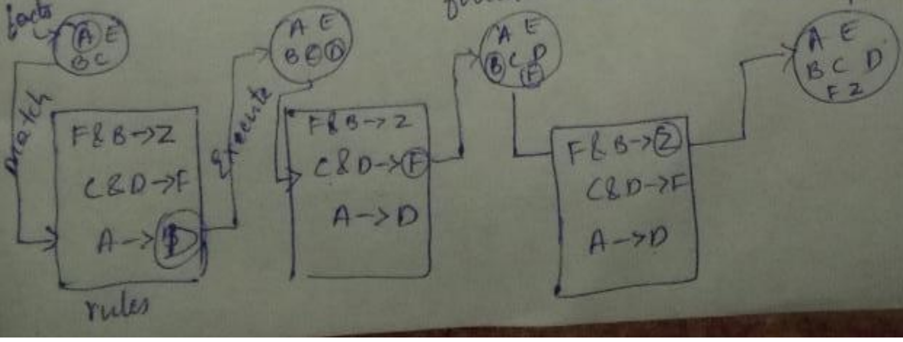
Pblm: To prove if A & B true, then D is true

TPT : A & B → D



Example 2: Goal State: Z

Termination condⁿ: stop if Z is derived (DB) or no further rules can be applied



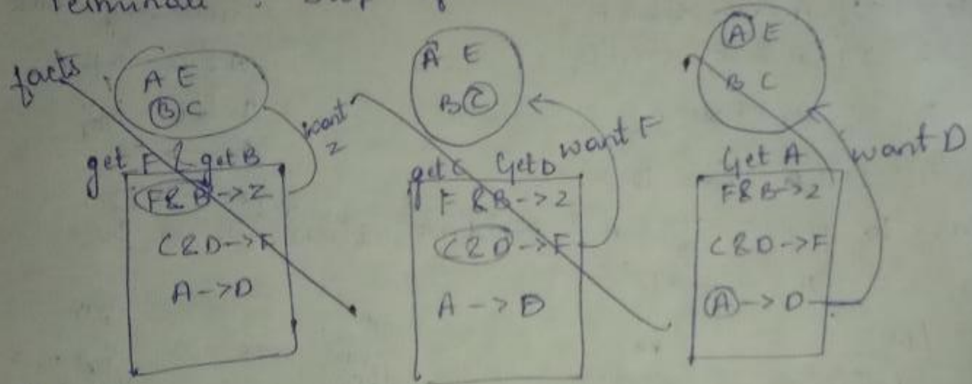
Backward Chaining

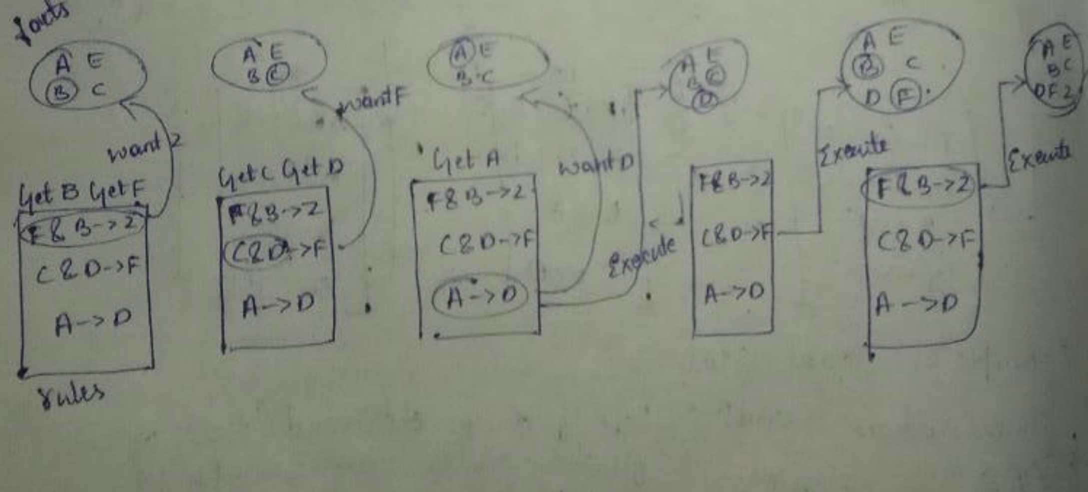
-> It is form of reasoning which starts with the goal & works backward, chaining thro rules to find facts support the goal

Properties

-> Top down approach is followed
-> It is based on modus pones inference rule
-> The goal is divided into sub goals
-> It is called goal driven approach, as list of goals decide which rules are selected & used
-> Used in game theory, assistants & ai appl^n
-> Used in DFS strategy for proof

Example Goal state is Z
Terminate : Stop if Z is derived





Lifting in fol:

????????????????????????