Unit-2: Knowledge Representation using FOPL

2.1. Knowledge Representation

In order to solve the complex problems encountered in AI, one needs both, a large amount of knowledge and some mechanisms for manipulating that knowledge to create solutions to new problems. A variety of ways of representing knowledge have been exploited in AI programs.

In KR, we deal with two different kinds of entities:

i. Facts i.e. things we want to represent.
ii. Representations of facts in some chosen formalism.

One such formalism is predicate logic.
Consider the English sentence: Spot is a dog.
The fact represented by that English sentence can be represented in predicate logic as: *dog(Spot)*.

2.1.1. Approaches to knowledge representation

A good system for KR in a particular domain should possess the following 4 properties:

i. Representational Adequacy: ability to represent all kinds of knowledge. ii.
Inferential Adequacy: ability to infer new knowledge from old.
iii. Inferential Efficiency: ability to incorporate additional information.
iv. Acquisitional Efficiency: ability to acquire/add new information easily.

2.1.2. Techniques for Knowledge representation

i. **Declarative**/Relational/Explicit Knowledge: provides facts e.g. employee database. ii.
Procedural/Implicit Knowledge: specifies what to do when e.g. rules, algorithm. iii. Inferential
knowledge: mechanism to infer new knowledge from existing e.g. resolution. iv. Inheritable
knowledge: mechanism to store common/basic objects separately.

2.1.3. Issues in Knowledge representation

i. Are any attributes of objects so basic that they occur in almost every problem domain? If there are, we need to make sure that they are handled appropriately in each of the mechanisms we propose.
ii. Are there any important relationships that exist among attributes of objects? iii.
At what level should knowledge be represented?
iv. How should sets of objects be represented?
v. Given a large amount of knowledge, how can relevant parts be accessed when they are needed?

2.2. Propositional Logic

A **proposition** is a statement which can be said to be either True or False.
A proposition is represented by a propositional variable e.g. p: Today is Sunday. Here, “p” is a propositional variable representing the proposition “Today is Sunday”. A propositional variable can have one or more letters/characters.

A compound proposition consists of more than one propositions connected by logical operators.

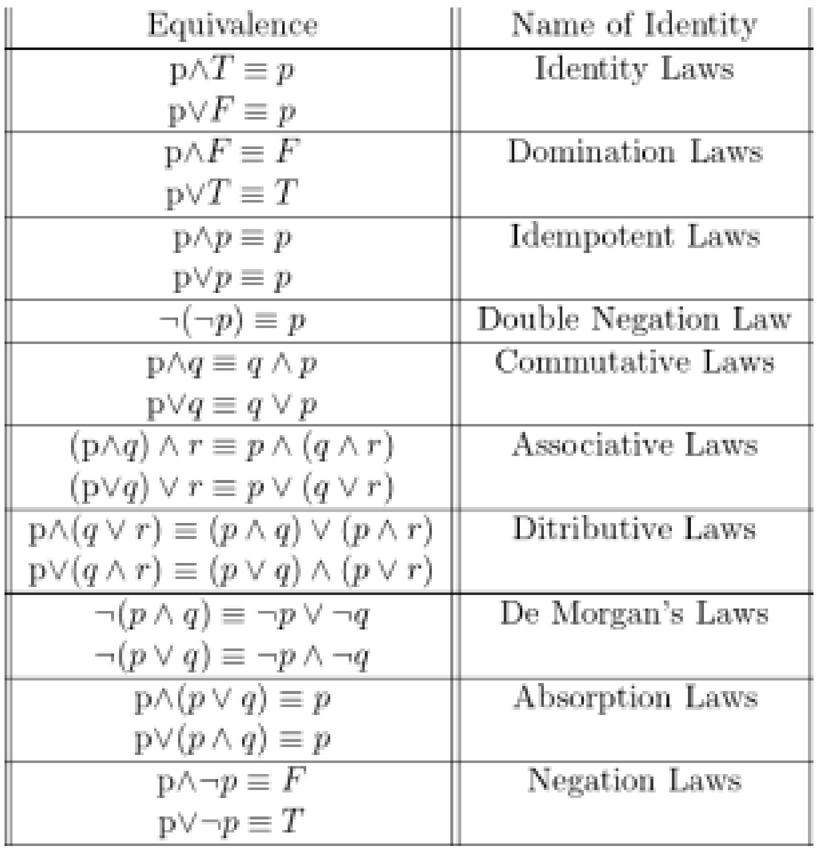
Logical operators used with proposition are:

- Conjunction: corresponding to AND, symbol is ∧
- Disjunction: corresponding to OR, symbol is ∨
- Negation: corresponding to NOT, symbol is ¬ or ~
- Implication: corresponding to IF-THEN, symbol is ⟶
- Bi-implication: corresponding to bi-directional IF-THEN, symbol is⟷

For example, the sentence "If today is Sunday then today is a holiday" can be represented as p⟶q, where "p: Today is Sunday" and "q: Today is a holiday".

A **well-formed formula (wff)** is the representation of a compound proposition using propositional variables and logical operators.
A **literal** is a propositional variable in its positive or negative form.

Truth tables give the truth value of compound propositions based on all possible combinations of propositions being True or False.
If a compound proposition is True for all such combinations then it is called a **tautology** e.g. p∨¬p. If a compound proposition is False for all such combinations then it is called **contradiction** e.g. p∧¬p.
Two wff's are said to be **equivalent (≡)** if they have the same truth table e.g. p⟶q≡¬p∨q.



Equivalence	Name of Identity
$p \wedge T \equiv p$ $p \vee F \equiv p$	Identity Laws
$p \wedge F \equiv F$ $p \vee T \equiv T$	Domination Laws
$p \wedge p \equiv p$ $p \vee p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \wedge q \equiv q \wedge p$ $p \vee q \equiv q \vee p$	Commutative Laws
$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$ $(p \vee q) \vee r \equiv p \vee (q \vee r)$	Associative Laws
$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	Ditributive Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's Laws
$p \wedge (p \vee q) \equiv p$ $p \vee (p \wedge q) \equiv p$	Absorption Laws
$p \wedge \neg p \equiv F$ $p \vee \neg p \equiv T$	Negation Laws

2.2.1. Inferences

Propositional Logic allows us to make inferences based on rules of inference.
Each rule of inference is a tautology.

Name of Rule	wff	Example
Modus Ponens	[(p⟶q) ∧ p] ⟶ p	If today is Sunday then today is a holiday. Today is Sunday.

		Therefore, today is a holiday.
Modus Tollens	[(p⟶q) ∧ ¬q] ⟶ ¬p	If today is Sunday then today is a holiday. Today is not a holiday. Therefore, today is a not Sunday.
Hypothetical Syllogism	[(p⟶q) ∧ (q⟶r)] ⟶ (p⟶r)	If today is Sunday then today is a holiday. If today is holiday then I will watch movie. Therefore, if today is Sunday then I will watch movie.
Disjunctive Syllogism	[(p∨q) ∧ ¬p] ⟶ q	Today is either Saturday or Sunday. Today is not Sunday. Therefore, today is Saturday.
Resolution	[(p∨q) ∧ (¬p∨r)] ⟶ (q∨r)	I will play or I will study. I will not play or I will watch movie. Therefore, I will study or watch movie.

The following forms are fallacies (incorrect form of reasoning):

- [(p⟶q) ∧ q] ⟶ p
- [(p⟶q) ∧ ¬p] ⟶ ¬q

2.2.2. Forward vs Backward Reasoning

In forward chaining we move from start state to goal state.
In backward chaining, we move from goal state to start state.
Let suppose, we have A⟶B, B⟶C, C⟶D, D⟶E, D⟶F, D⟶G; start state is A and goal is G. As A is start state, in forward chaining we go from A to B, B to C, C to D, D to E, D to F, D to G. As G is goal state, in backward chaining, for G to be true, D has to be true; for D to be true, C has to be true; for C to be true, B has to be true; for B to be true, A has to be true. As A is true, B, C, D, G are true.

# Prolog does backward chaining.

2.3. Predicate Logic/FOPL

Predicate logic helps us to reason with generic statements like "All humans are mortal", "Some men are intelligent", etc.
In predicate logic, we identify the subject(s) and predicate/relation from the statement and write in the form **predicate(subject)** or **relation(subject-1,subject-2)**.
Predicate logic provides two additional logical operators known as **quantifiers**: universal quantifier (represented by ∀) and existential quantifier (represented by ∃). Universal quantifier is used when the given predicate is true for all subjects, whereas existential quantifier is used when the given predicate is true for some subjects but not for all. With each quantifier a variable is associated which is used as unknown subject.

Sachin is intelligent.	intelligent(Sachin)
Everyone is intelligent.	∀x: intelligent(x)

All humans are intelligent.	∀x: human(x) ⟶ intelligent(x)
Some men are intelligent.	∃y: man(x) ∧ intelligent(x)
John likes peanuts	likes(John,peanuts)
John likes everyone.	∀x: likes(John,x)
John likes someone.	∃y: likes(John,y)
Everyone likes someone.	∀x: ∃y: likes(x,y)

NOTE: implication is used with universal quantifier but not with existential quantifier.

EXAMPLES

Arts courses are easy.	∀x: Arts_course(x) ⟶ easy(x)
Steve likes easy courses.	∀x: easy(x) ⟶ likes(Steve,x)
John likes all kinds of food.	∀x: food(x) ⟶ likes(John,x)
Apples are food.	food(apple)
AK-301 is an Arts course.	Arts_course(AK-301)
Sue eats everything Bill eats.	∀x: eats(Bill,x) ⟶ eats(Sue,x)
Everything someone eats and is not killed by is food.	∀x∃y:eats(y,x)∧¬killedby(y,x)⟶food(x)

FOPL (First-Order Predicate Logic): if a subject has more than one predicates then they must be specified separately and connected by conjunction e.g. girl(Jen) ∧ intelligent(Jen). HOPL (First-Order Predicate Logic): allows nesting of predicates e.g. intelligent(girl(Jen)).

2.3.1. Natural Deduction

In natural deduction, we move from generalization to specialization i.e. something which is true for all will also be true for any specific subject. For example, given that “All humans are mortal. I am a human”, it can be deduced that “I am mortal”.

Universal Instantiation: If P is true for all x, P is true for c.
Existential Instantiation: If P is true for some x, P is true for c.

Universal Generalization/Introduction: If P is true for some c, P is true for all x.
Existential Generalization/Introduction: If P is true for c, P is true for some x.

2.3.2. Unification

In predicate logic, man(John) and ¬man(john) is a contradiction but man(John) and ¬man(Spot) is not. Thus in order to determine contradictions we need a matching procedure that compares two literals and discovers whether there exists a set of substitutions that makes them identical.

Rules:-
To attempt to unify two literals, we first check if their initial predicate symbols are the same.
If the predicate symbols match then we check the arguments, one pair at a time. Different constants and predicates cannot match; identical ones can.
A variable can match another variable, any constant or a predicate.

f(x) and g(x) can't be unified as predicates are different.
f(Marcus) and f(Caesar) can't be unified as Marcus and Caesar are constants.
f(Marcus) and f(x) can be unified as f(x/Marcus).

Q. Can f(Marcus,g(x,y)) and f(x,g(Caesar,Marcus)) be unified?
Answer.
The predicate is same i.e. "f".
The first pair of arguments is Marcus and x, which can be unified, so x is substituted with Marcus. The second pair of argument is a predicate which is same i.e. "g". Inside this predicate, we take the first pair of arguments i.e. x and Caesar. We know, x is already substituted with Marcus, so it can not be substituted with another constant. So, unification fails.

2.3.3. Clausal Form

Clausal form allows only disjunction and negation, rest all the logical operators have to be removed from wff.

Steps to conversion into clausal form

i. Remove implications: replace p⟶q by ¬p∨q.
ii. Move negation inwards (DeMorgan law)
iii. Standardize variables: each quantifier must be associated with a unique variable. iv.
Move quantifiers to the left (Prenex Normal Form)
v. Drop existential quantifiers (Skolemization): replace ∃yP(y) with P(k) vi.
Drop universal quantifiers
vii. Apply distributive law to convert to CNF
viii. Drop conjunctions

A **horn clause** is a clause with at most one positive literal.

Q. Convert following into clausal form: ∀x∃y:eats(y,x)∧¬killedby(y,x)⟶food(x).
ANSWER
Remove implication, ∀x ¬[∃y:eats(y,x)∧¬killedby(y,x)] ∨ food(x)
Move negation inwards, ∀x∀y¬eats(y,x) ∨ killedby(y,x) ∨ food(x)
Drop universal quantifiers, ¬eats(y,x) ∨ killedby(y,x) ∨ food(x)

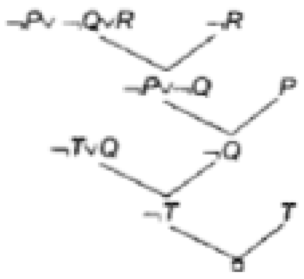
2.3.4. Resolution in Predicate Logic

Resolution is a method of proof. It is also known as proof by refutation/contradiction i.e. to prove P is true, we show that ¬P is False.

Resolution only works on clauses, so we need to convert all the wff's into clauses.

Given the clauses:
P
¬P∨¬Q∨R
¬S∨Q
¬T∨Q
T
To prove R.
We will negate the clause to be proved i.e. we will take the clause ¬R.

In resolution, we take two clauses at a time, one having a positive literal and other negative literal; apply resolution to generate intermediary clause, and continuing the process. The goal is to achieve NULL in the end.



EXERCISE

Q1. Assume the following facts:

- Steve likes easy courses
- Arts courses are easy.
- Science courses are hard
- AK-301 is an Arts course.

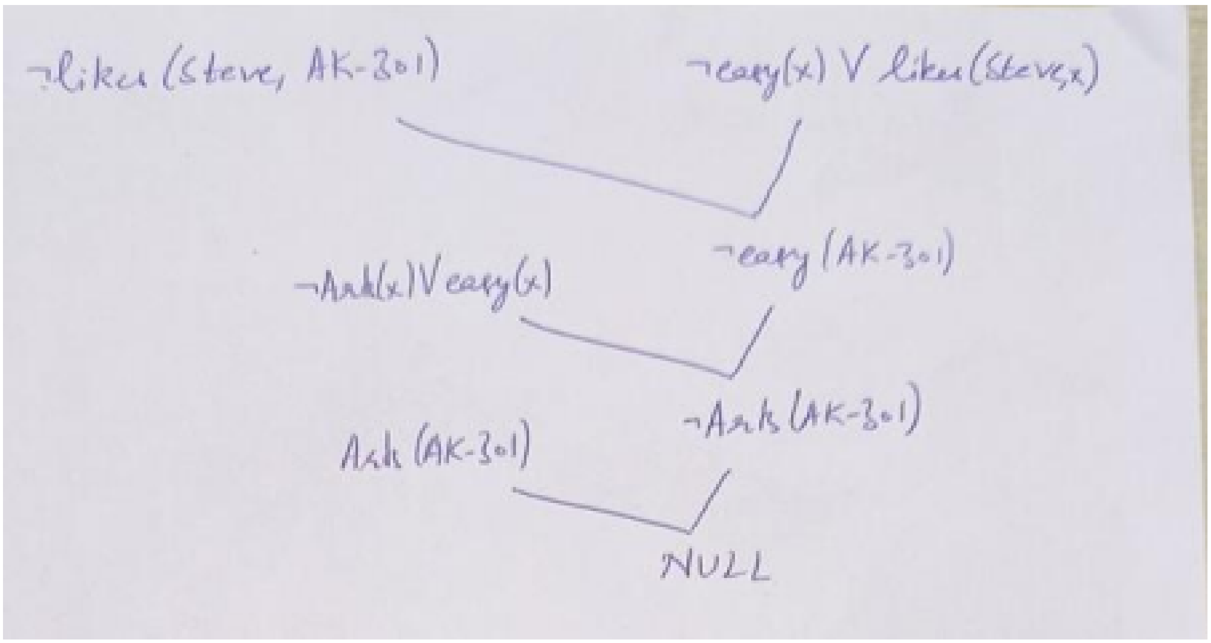
Use resolution to answer the question "What course would Steve like?".

ANSWER
As we are provided with only one course i.e. AK-301 here, we will check whether Steve likes AK-301.

FOPL
∀x: easy(x) ⟶ likes(Steve,x)
∀x: Arts(x) ⟶ easy(x)
∀x: Science(x) ⟶ ¬easy(x) [as hard is opposite of easy] Arts(AK-301)

CLAUSAL FORM
¬easy(x) ∨ likes(Steve,x)
¬Arts(x) ∨ easy(x)
¬Science(x) ∨ ¬easy(x)
Arts(AK-301)
¬likes(Steve,AK-301)

RESOLUTION
We add the clause, ¬likes(Steve,AK-301) i.e. negation of what is to be proved.



Q2. Consider the following sentences:

- John likes all kinds of food.
- Apples are food.
- Everything someone eats and is not killed by is food.
- Bill eats peanuts and is still alive.
- Sue eats everything Bill eats.

(a) Prove John likes peanuts with backward chaining.
(b) Prove John likes peanuts with resolution.

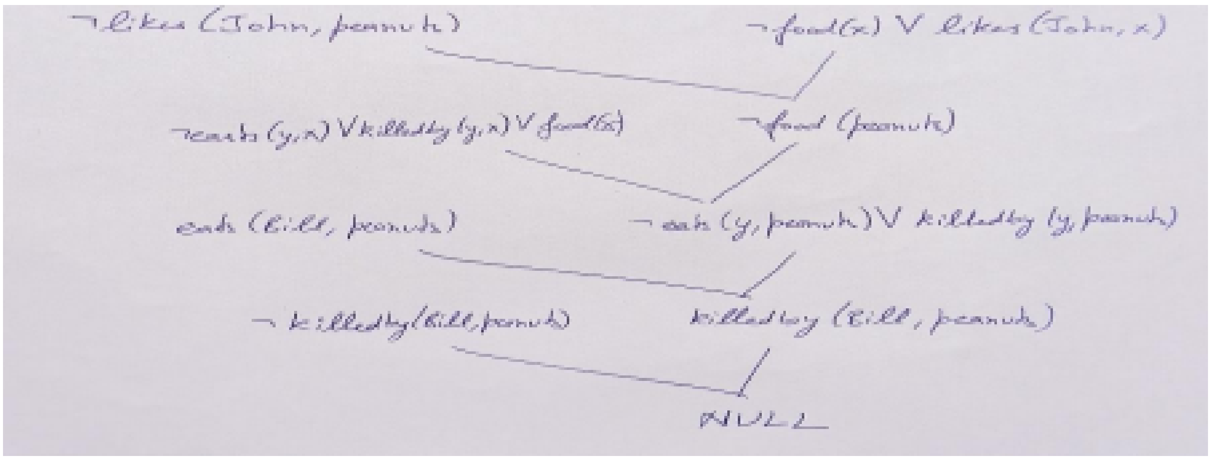
ANSWER
To prove: likes(John, peanuts)

FOPL
∀x: food(x) ⟶ likes(John,x)
food(apple)
∀x∃y:eats(y,x)∧¬killedby(y,x)⟶food(x)
eats(Bill,peanuts) ∧ ¬killedby(Bill,peanuts) [alive is same as ¬killedby] ∀x:eats(Bill,x) ⟶ eats(Sue,x)

BACKWARD CHAINING
Goal: likes(John, peanuts)
∀x: food(x) ⟶ likes(John,x)
Goal: food(peanut)
∀x∃y:eats(y,x)∧¬killedby(y,x)⟶food(x)
Unifying this with
eats(Bill,peanuts) ∧ ¬killedby(Bill,peanuts)
we get food(peanut)

CLAUSAL FORM
¬food(x) ∨ likes(John,x)
food(apple)
¬eats(y,x) ∨ killedby(y,x) ∨ food(x)
eats(Bill,peanuts)
¬killedby(Bill,peanuts)
¬eats(Bill,x) ∨ eats(Sue,x)

RESOLUTION
We add the clause: ¬likes(John,peanuts)



Q3. Represent the following facts as predicates:
i. Some people like football.
ii. Riya and Siya are sisters.
iii. Every student smiles.
iv. There is an Anaconda that is poisonous than all Pythons.
v. Someone walks and someone talks.

ANSWER
∃x: person(x) ∧ likes(x,football)
sisters(Riya,Siya)
∀x: student(x) → smiles(x)
∀x: Python(x) → ∃y:Anaconda(y) ∧ more_poisonous(y,x)
∃x: walks(x) ∨ talks(x)

Q4. Check the validity of the following argument:
If he gets the job and work hard, then he will get promoted. If he gets promoted, then he will be happy. He is not happy. Therefore, either he did not get the job or he did not work hard.

ANSWER
J: he gets the job
W: he works hard
P: he gets promoted
H: he is happy
J ∧ W → P
P → H
¬H
According to modus tollens, [(P→H) ∧ ¬H] → ¬P
Similarly [(J ∧ W → P) ∧ ¬P] → ¬(J ∧ W)

¬(J ∧ W) gives ¬J ∨ ¬W which means “He did not get the job or he did not work hard”.
So given conclusion is correct.

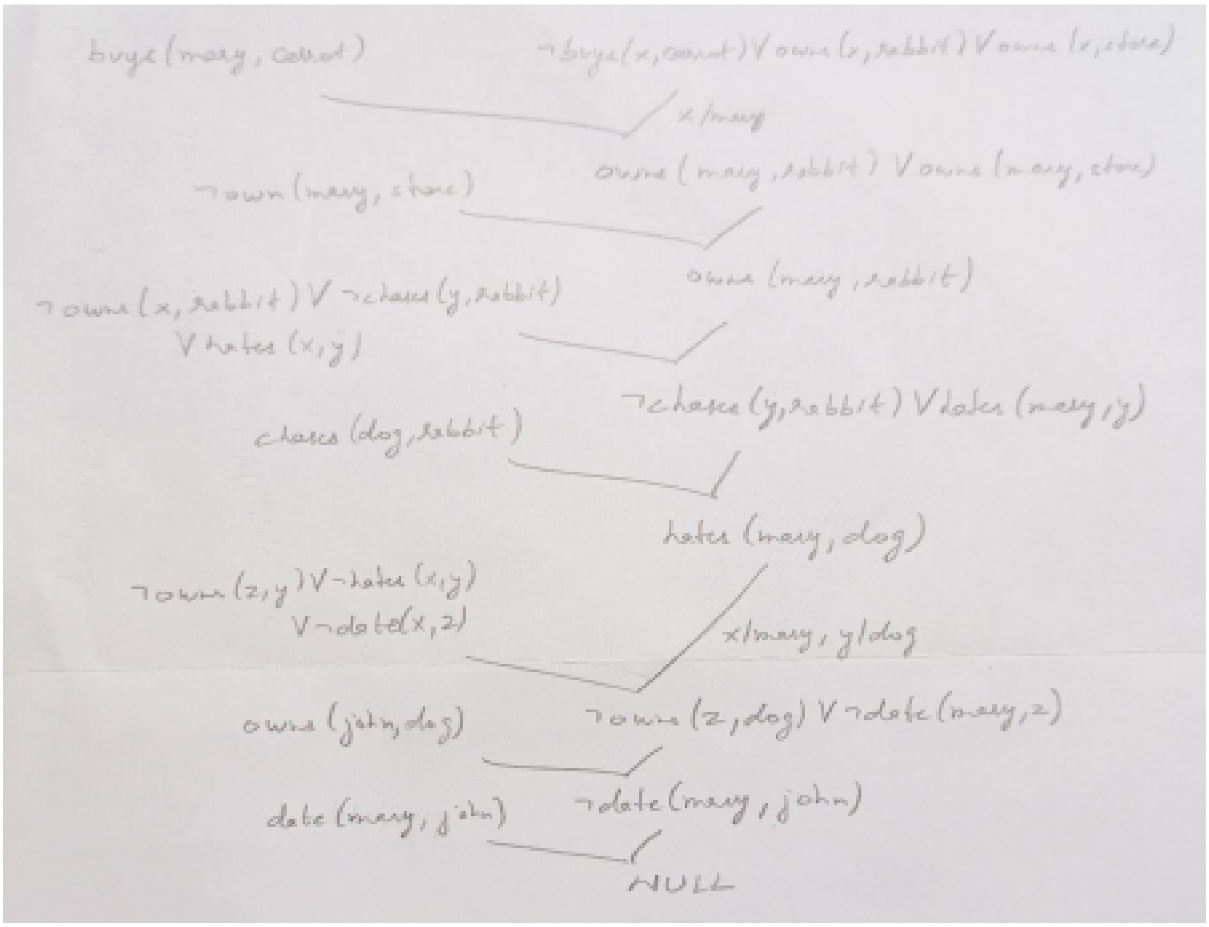
Q5. Prove the validity of the following argument:
All metro cities have huge traffic. Delhi has huge traffic. Therefore, Delhi is a metro city.
ANSWER
[∀x: metro_city(x) → huge_traffic(x)] ∧ huge_traffic(Delhi) → metro_city(Delhi) However, correct rules of inference are [(P→Q) ∧ P] → Q and [(P→Q) ∧ ¬Q] → ¬P. As, the given inference does not match either modus ponens or modus tollens, therefore the given conclusion is incorrect.

Q6. Convert the following into FOL

- Anyone who buys carrot own either a rabbit or a grocery store.
- Every dog chases some rabbit.
- Mary buys carrot.
- Anyone who owns a rabbit hates anything that chases any rabbit.
- John owns a dog.
- Someone who hates something owned by another person will not date that person. Convert above in clause form and prove by resolution following conclusion:- “If Mary does not own a grocery store, she will not date John”.

ANSWER
FOPL
Mary buys carrot. **buys(mary,carrot)**
John owns a dog. **owns(john,dog)**
Anyone who buys carrot own either a rabbit or a grocery store.
∀x:buys(x,carrot)→owns(x,rabbit) ∨**owns(x,store)**
Anyone who owns a rabbit hates anything that chases any rabbit.
∀x∃y: owns(x,rabbit) ∧ **chases(y,rabbit)** → **hates(x,y)**
Every dog chases some rabbit. **chases(dog,rabbit)**
Someone who hates something owned by another person will not date that person. **∀x∀z∃y: owns(z,y)** ∧ **hates(x,y)** → **¬date(x,z)**
Mary does not own a grocery store.
¬own(mary,store)
To Prove: **¬date(mary,john)**)

CLAUSAL FORM
buys(mary,carrot)
owns(john,dog)
¬buys(x,carrot) ∨ owns(x,rabbit) ∨ owns(x,store)
¬owns(x,rabbit) ∨ ¬chases(y,rabbit) ∨ hates(x,y)
chases(dog,rabbit)
¬owns(z,y) ∨ ¬hates(x,y) ∨ ¬date(x,z)
¬own(mary,store)
date(mary,john)


buys(mary, carrot)
¬buys(x, carrot) V owns(x, rabbit) V owns(x, store)
x/mary
¬own(mary, store)
owns(mary, rabbit) V owns(mary, store)
owns(mary, rabbit)
¬owns(x, rabbit) V ¬chases(y, rabbit) V hates(x, y)
¬chases(y, rabbit) V hates(mary, y)
chases(dog, rabbit)
hates(mary, dog)
¬own(z, y) V ¬hates(x, y) V ¬date(x, z)
x/mary, y/dog
owns(john, dog)
¬owns(z, dog) V ¬date(mary, z)
date(mary, john)
¬date(mary, john)
NULL

Q7. Consider the following facts:

• The members of Elm St Bridge Club are Joe, Sally, Bill, and Ellen. •
Joe is married to Sally.
• Bill is Ellen's brother.
• The spouse of every married person in the club is also in the club. •
The last meeting of the club was at Joe's house.

Represent these facts in predicate logic and convert to clausal form.

ANSWER

FOPL
member(Joe,ESBC) ∧ member(Sally,ESBC) ∧ member(Bill,ESBC) ∧
member(Ellen,ESBC) married(Joe, Sally)
brother(Bill, Ellen)
∀x∀y: married(x,y) ∧ member(x,ESBC) → member(y,ESBC)
meeting(last,Joe)

CLAUSES
member(Joe,ESBC)
member(Sally,ESBC)
member(Bill,ESBC)
member(Ellen,ESBC)

married(Joe, Sally)
brother(Bill, Ellen)
¬married(x,y) ∨ ¬member(x,ESBC) ∨ member(y,ESBC)
meeting(last,Joe)

∧∨¬→∀∃