Predicate Logic

Dr. Priyadarshan Dhabe
Ph.D (IIT Bombay), Assistant Professor,
Department of Information Technology

What is a Logic?

Logic:- is a **science of reasoning**. It defines **representation**, **Manipulation** and **uses** of real world facts. (Knowledge).

Logic:-The formal, systematic study of the principles of **valid inference** and **correct reasoning** (Ref:-Penguin Encyclopedia)

Aristotle –pioneered this field by codifying "Right Thinking"

Components of a Logic System

- Syntax Rules- how to represent a fact in a valid manner using symbols?
- Semantic Rules- How to interpret a represented fact?
- Inference Rules- How to generate new and valid facts from existing facts?
- A set of symbols- Used to represent real world facts.

Properties of Logic systems

- Consistency :- No theorem (proof) of the system contradicts to another.
- Soundness:-which means that the system's rules of proof will never allow a false inference from a true premise (basis/ argument). If a system is sound and its axioms are true then its theorems are also guaranteed to be true.
- Completeness:-which means that there are no true sentences in the system that cannot, at least in principle, be proved in the system.

Propositional Logic

- A Logic system already Known to you. Also called as sentential logic.
- Proposition- is a sentence having truth values either “True” or “False”.

Problems with Propositional Logic(PL).

- Assume following propositions.
  - Tommy is a dog. Represented with “P”
  - Moti is a dog. Represented with “Q”
- Consider these representations only, P and Q. It is very difficult to find the commonality between “Tommy” and “Moti”.
- Lack of expressiveness. Not every real world fact can be expressed in PL concisely (brief).
- Representation lack focus on objects and their relationship.

Predicate Logic Introduction

Also called as "**First order logic**", "**Quantificational logic**", "**First order predicate calculus**"-wikipedia

- **First order refers to arguments of predicate are objects e.g red(Ball)**
- **(Red- predicate, Ball is an object)**
- **Higher order logic- arguments of a predicate can be predicates**

Predicate Logic Introduction

- **What is a predicate?**
- **Any sentence describes objects and their properties, classes and relationships.**
- **Predicate is a part of sentence describing properties, classes and relationships of objects.**
- **E.g. This Ball is Red.**
- **(Red- predicate, Ball is an object)**

Representing Facts in Predicate Logic (informal way)

- Write predicate outside the bracket ( ) and objects inside the bracket.
- This ball is red.

  red (Ball). **unary**
- ***Ram is brother of Sham***

  brother (Ram, Sham) **binary**
- ***Sham defeated Hari and Ram***

  defeated(Sham, Hari, Ram) **Ternary**
- A predicate can have n objects or of **n-arity**

  n>=0.

General form

pqrst (A,B, C)

Relationship, Class

Or

Property

Objects

involved

Represent following simple facts in predicate logic

1. P. S. Dhabe is assistant professor.
2. Students appeared for exam.
3. Train is late.
4. A and B played chess.
5. Sky is blue.
6. A, B and C are project group members.

Relationship between Proposition and Predicate

Proposition= Predicate+ Objects involved in the fact

Components Of Predicate Logic

- Set of Objects (Object Constants)
- Set of functions (Function Constants)
- Set of Relations (Relation Constants)
- Connectives ($\wedge, \vee, \neg$)
- Delimiters (,), [,], seperator ,
- Quantifiers ($\forall$ $\exists$)

Have a closer look at Syntax and Semantics.

Syntax of Objects

- These are alphanumeric strings.
- They either starts with a Capital letter or a number.

e.g. EifelTower,
Exam,
12box,
Ab,

Are valid Object representations in (PL)

Semantics of Objects

- They can physical objects like *box* or *pen*.
- They can be abstract entities like number 7 & Л.
- They can be fictional entities like- "*Santa Claus*"

Syntax of Functions

- These are alphanumeric strings always begins with a *Capital letter*.
- These are the computable functions allowed to be used and called as "*Skolem Functions*", due to logician *Thoralf Skolem* [1920].

e.g *Father(x),* x- object variable

This function returns name of the father of x so that we can write

congratulated(y, Father(x))

Semantics of Functions

A function can be of n-arity and it maps n objects into another objects.

e.g *Childs(y)* will return all the child names of object y.

Note:-Typographical conventions used here are not universal.

Syntax of Relations

- Relation can be-
  - property and object-Class relationship.
- These are alphanumeric strings always begin with ***small case letters***.
- A relation constant is also called as a "predicate". (Formal Definition)

defeated(Ram, Sham)

Semantics of Relations

- Each relation will have **arities**.
- Objects can participate in **arbitrary number** of relations.
- A relation with **arity 1** is called as "***property***" or A "***object-class***" relationship.

  red(Ball) ***bird(crow)***
- They can have **zero objects**.

  e.g. There is raining.

  raining( ) or simply ***raining***

Syntax Rules in predicate Logic

Talks about ***"How to write a valid WFF?".***

1. **True** and **False** (represented as **T** and **F**) are valid WFF.

   (They are called as **distinguished Atoms**).

2. **Atom** or **Atomic formula** is a valid WFF.

   "***A predicate followed by n objects (n>=0) in parenthesis separated by comma***" is called as ***Atom*** or ***Atomic formula***.

   playedchess(A, B) – is "***Atomic***" Formula

   playedchess(A, B)^ winner(B) – "***Composite***" Formula

Syntax Rules in predicate Logic

3. If R_1 and R_2 are valid WFF then following are also WFF

$a).R_1 \wedge R_2 \quad (Conjunction)$

$b).R_1 \vee R_2 \quad (disjunction)$

$c).\neg R_1 \quad (Negation)$

$d).R_1 \rightarrow R_2 \quad (implication)$

$e).A\ WFF\ involving\ Quantifiers\ is\ also\ a\ valid WFF$

$\forall x : drive(x) \qquad \forall y : \exists x : drive(x)$

4. There are no other WFF

Truth Table for Implication

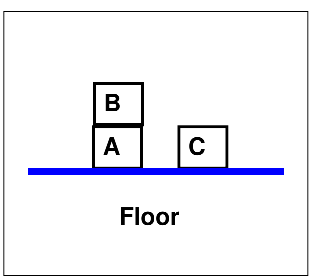
A	B	A→B
T	T	T
T	F	F
F	T	T
F	F	T

What is "*interpretation*" of a formula?

Predicate calculus formulas (PCF) and Knowledge

- PCF can be used to represent "***Knowledge***" that an ***agent*** has about the world.
- ***Knowledge base***:- A set δ of such wffs is called as knowledge base.
- If a formula w is in δ then we say that agent "***Knows***" w, or more appropriately agent "***believes***" in w.

Example of Knowledge base



Blocks World Domain

On (A, Floor)

On (B, A)

On (C, Floor)

clear (B)

clear (C)

$$on(B, A) \rightarrow \neg clear(A)$$

Knowledge base

Quantification

- Is a process that specify "***quantity***" of objects involved in a Fact in abstract/ general way. Mostly "***at least one***" or "**all**".
- This can be done using "***Quantifiers***".
- There are two **types** of Quantifiers
  - **Universal Quantifier.**
  - **Existential Quantifier.**

Universal Quantifiers (UQ) $\forall$

- Suppose we want to say that ***every object*** in the **domain** has a **certain property** or **participated** in a relation. If the **domain is finite** (3dogs A, B and C), we want to represent "***every dog has a tail***" . It can be represented using **conjunction** as

$$dog(A) \rightarrow hasatail(A) \wedge$$
$$dog(B) \rightarrow hasatail(B) \wedge$$
$$dog(C) \rightarrow hasatail(C)$$

- But if the domain is ***infinite*** we can not list all of them
- Such situation is handled with a (UQ)
- A UQ is used to represent "***for all***" and "***every***" phrases.

$$\forall x : dog(x) \rightarrow hasatail(x)$$

Syntax rule using UQ

$$\begin{aligned}&if\ \ w(x)\ is\ a\ WFF\\&\ then\\&\forall x: w(x)\ is\ also\ a\ WFF\end{aligned}$$

Where x-is a object variable
NOT an Object

"Truth value" of wff with Universal quantifiers

$$\forall x: dog(x) \rightarrow hasatail(x)$$

Above WFF is "TRUE" if it has a value "TRUE" for <u>all the objects</u> x in that domain.

Existential Quantifier $\exists$

- Suppose we want to say that "*at least one*" object has a certain property. It can be handled using "disjunction" in finite domain.
- Assume a finite domain of 3 persons A, B and C. We want to represent a fact " *at least one person can drive*".

$$drive(A) \vee drive(B) \vee drive(C)$$

- In infinite domain an Existential Quantifier is used

$$\forall x : \exists y : drive(y)$$

Set-subset relationship?

"Truth value" of wff with Existential quantifiers

$$\forall x : \exists y : drive(y)$$

Above WFF is "TRUE" if it has a value "TRUE" for "at least one object" x in that domain.

Some Rules about Quantifiers

1. $(\forall x):[(\forall y):w]$ *is equivalent to* $(\forall y):[(\forall x):w]$
   We can change order of universal Quantifiers.
2. $(\forall x):[(\forall y):w]$ *is equivalent to* $\underbrace{(\forall x):(\forall y)}_{\text{prefix}}:\underbrace{[w]}_{\text{Matrix}}$
   We can group universal Quantifiers.
3. *Rule*1 *and Rule*2 *are also applicable to Exsitential Quantifiers*
4. $(\forall x):[(\exists y):w]$ *is not equivalent to* $(\exists y):[(\forall x):w]$
   grouping of universal and Exsitential Quantifier must retain their order

Some Rules about Quantifiers

5. $(\forall x):[w]$ *is equivalent to* $(\forall y):[w]$
   if all occurances of x are replaced by y
   We can change variable name.
   (applicable to EQ also)
6. $\neg(\forall x):w$ *is equivalent to* $(\exists x):\neg w$
   $\neg(\exists x):w$ *is equivalent to* $(\forall x):\neg w$
   Change of Quantifiers with reducing scope of $\neg$.

Inference Rules in Predicate Logic

1. ***Modus-ponen*** **(LATIN mode that affirms/state/asserts)**

If P and P→Q are given

Then infer Q

$If\ \ dog(Tommy)\ and\ \forall x: dog(x) \rightarrow hasatail(x)\ Then$

$INFER$

$hasatail(Tommy)$

Inference Rules in Predicate Logic

2. (AND ^ Introduction)

IF P and Q are given
then Infer

P^Q

$If\ \ dog(Tommy)\ and\ \ hasatail(Tommy)\ are\ \ given$

$Then$

$INFER$

$dog(Tommy) \wedge hasatail(Tommy)$

Inference Rules in Predicate Logic

3. (Commutativity of ^)

IF P ^ Q is given then Infer

Q^P

Inference Rules in Predicate Logic

4. (^ Elimination)

IF P ^ Q is given then Infer

(either Q) or (either P)

Inference Rules in Predicate Logic

5. (V Introduction)

IF P is given

then Infer

P v Q

If dog(Tommy) is given

Then

INFER

$dog(Tommy) \vee black(Tommy)$

Inference Rules in Predicate Logic

6. (¬ Elimination)

IF ¬(¬P) is given

then Infer

P

Inference Rules in Predicate Logic

7. (Universal Instantiation UI)

$If\ \forall x : w(x)\ is\ given$

$Then\ INFER$

$w(\alpha)\ where\ \alpha\ is\ a\ real\ object\ from\ a\ domain$

$(replace\ all\ x\ by\ \alpha\ in\ that\ WFF)$

$If\ \ \forall x : dog\ (x) \rightarrow hasatail\ (x)\ is\ given$

$Then\ \ INFER$

$dog\ (Tommy\ ) \rightarrow hasatail\ (Tommy\ )$

General to specific

Inference Rules in Predicate Logic

8. (Existential Generalization EG)

$If\ \ w(\alpha)\ is\ given$

$Then\ INFER$

$\exists x : w(x)\ \ where\ \alpha\ is\ a\ real\ object$

$from\ a\ domain\ \&\ x\ is\ object\ Variable$

$INFER\ \exists x : dog(x)$

$if\ \ dog(Tommy)\ is\ given$

Specific to General

ISA Hierarchy

- How to represent Class- subclass and instance relationships Explicitly?
- Isa- predicate is used to represent Class- subclass relationship like.

isa (Mammal, Animal)
isa (Human, Mammal)

Not Objects

- Arguments to the isa predicate are "Class labels".
- Instance-predicate is used to represent object and class relationship *explicitly*.

instance (Tommy, Dog)

- Arguments to the instance predicate are object and class label.

Representing more facts in Predicate Logic

1. Marcus was a man.

$man(Marcus)$ (Tense is ignored)

2. Marcus was Pompeian.

(Not:-Pompeii, an ancient city of southern Italy southeast of Naples. Founded in the sixth or early fifth century b.c., it was a Roman colony by 80 b.c. and became a prosperous port and resort with many noted villas, temples, theaters, and baths. Pompeii was destroyed by an eruption of Mount Vesuvius (volcano) in a.d. 79.)

$\text{pompeian}(\text{Marcus})$

3. All Pompeian's were Romans.

$$\forall x : pompeian(x) \rightarrow roman(x)$$

4. Caesar was a ruler

$$ruler(Caesar)$$

Representing more facts in Predicate Logic

5. All Romans were either loyal to Caesar or hated him.

$$\forall x : roman(x) \rightarrow loyalto(x, Caesar) \vee hate(x, \text{Caesar})$$

6. Everyone is loyal to someone.

$$\forall x : \exists y : loyalto\ (x, y)$$

7. People only try assassinate rulers they are not loyal to.

$$\forall x : \forall y : person(x) \wedge ruler(y) \wedge tryassas\sin ate(x, y) \rightarrow \neg loyalto(x, y)$$

8. Marcus tried to assassinate Caesar.

$$tryassas\sin ate(Marcus, Caesar)$$

Representing more facts in Predicate Logic

9. Every man is a person.

$$\forall x : man(x) \rightarrow person(x)$$

Suppose we want to prove that "***Marcus was not loyal to Caesar***". A Human expert can prove this in following way.

$$\neg loyalto(Marcus, Caesar)$$

$$\uparrow \quad (7, sustitution)\ x/Marcus \quad y/Caesar$$

$$person(Marcus) \wedge \cancel{ruler(Caesar)} \wedge tryassas \sin ate(Marcus, Caesar)$$

$$\uparrow \quad (Applying\ \ axiom\ 4)$$

$$person(Marcus) \wedge tryassas \sin \cancel{ate(Marcus}, Caesar)$$

$$\uparrow \quad (Applying\ \ axiom\ 8)$$

$$person(Marcus)$$

$$\uparrow \quad (9\ substitution)\ x/Marcus$$

$$\cancel{man(Marcus)}$$

$$\uparrow \quad (Applying\ \ fact\ 1)$$

NULL

Problems in automating the proofs

- **Suppose a machine has to prove a statement that "*Marcus was not loyal to Caesar*" and having knowledgebase represented using WFF in predicate logic. e.g a WFF**

$$\forall x : \forall y : person\ (x) \wedge ruler\ (y) \wedge tryassas\ \ \sin ate(x, y) \rightarrow \neg loyalto\ (x, y)$$

Problems

1.System will not be efficient due to "Conjunctions"

2. Conjunction forces system to think more and thus requires more efforts.

3. Quantifiers add-up the same effect.

Solution :- A clause form

What is a Clause?

- A clause is fact represented without "Conjunction" and "Quantifiers".
- A fact represented using single WFF in predicate logic can be converted into multiple clauses.
- These Clauses are called "Horn Clauses"

(Logician Alfred Horn, 1951)

Converting a Fact into clause form.

Rule-1. Convert implication into their equivalent using rule

$$a \rightarrow b \equiv \neg a \vee b$$

"All Romans who know Marcus either hate Caesar or think that anyone who hates anyone is a crazy".

$$\forall x : [roman(x) \wedge know(x, Marcus)] \rightarrow$$
$$[hate(x, Caesar) \vee (\forall y : \exists z : hate(y, z) \rightarrow thinkcrazy(x, y))]$$

$$\forall x : \neg[roman(x) \wedge know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\forall y : (\neg \exists z : hate(y, z)) \vee thinkcrazy(x, y))]$$

Converting a Fact into clause form.

Rule-2:- Reduce the scope of negation to a single term using the above rules

$$\neg(\neg p) \equiv p$$

$$\neg(a \wedge b) \equiv \neg a \vee \neg b$$

$$\neg(a \vee b) \equiv \neg a \wedge \neg b$$

deMorgan's Law

$$\neg \forall x : P(x) \equiv \exists x : \neg P(x)$$

$$\neg \exists x : P(x) \equiv \forall x : \neg P(x)$$

Standard correspondence between quantifiers

$$\forall x : \underbrace{\neg[roman(x) \wedge know(x, Marcus)]}_{} \vee$$
$$[hate(x, Caesar) \vee (\forall y : (\underbrace{\neg\exists z : hate(y, z)}_{}) \vee thinkcrazy(x, y))]$$

By applying Rule 2, above fact will be changed to

$$\forall x : [\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

deMorgan's law

Correspondence between quantifiers

Converting a Fact into clause form.

Rule-3:- Standardize variables so that each quantifier binds a unique variable. e.g.

$$\underbrace{\forall x : P(x)}_{} \vee \forall x : Q(x)$$

$$\forall x : P(x) \vee \forall y : Q(y)$$

Scope of x

$$\forall x : [\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

Rule 3- is not applicable to this fact since each quantifier binds a single variable.

Converting a Fact into clause form.

Rule-4:- Move all the quantifiers to the left of formula without changing their relative order. This is possible since there is no conflict among variable names.

$$\forall x : [\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\forall y : \forall z : \neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

After application of Rule-4 to above fact it will be changed to

$$\forall x : \forall y : \forall z : [\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

$$\underbrace{\forall x : \forall y : \forall z}_{\text{prefix}} : [\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

At this point formula is said to be in Prenex normal form, containing prefix of quantifiers followed by matrix.

Matrix

Converting a Fact into clause form.

Rule-5:- Eliminate existential quantifiers using "Skolem Functions" as follows.

$$\exists y : president(y)$$
$$replace\ by$$
$$president(s1)$$

Where s1 is a function with no arguments and that returns a value that satisfies president.

$$\forall x : \exists y : fatherof(y, x)$$
$$replace\ by$$
$$\forall x : fatherof(s2(x), x)$$

Converting a Fact into clause form.

Rule-6:- Drop the prefix. At this point all the remaining variables are universally quantified and any proof procedure can assume that each variable it sees is universally quantified.

After applying rule 6

$$[\neg roman(x) \vee \neg know(x, Marcus)] \vee$$
$$[hate(x, Caesar) \vee (\neg hate(y, z) \vee$$
$$thinkcrazy(x, y))]$$

Converting a Fact into clause form.

Rule-7:- Convert the matrix into conjunction of disjuncts (CNF) and remove the parenthesis. Following properties can be useful here.

$$a \vee (b \vee c) \equiv (a \vee b) \vee c$$ **Associative**

$$(a \wedge b) \vee c \equiv (a \vee c) \wedge (b \vee c)$$ **Distributive**

Since there is no "AND" operator in our case thus remove the parenthesis.

$$\neg roman(x) \vee \neg know(x, Marcus) \vee$$
$$hate(x, Caesar) \vee \neg hate(y, z) \vee$$
$$thinkcrazy(x, y)$$

Converting a Fact into clause form.

Rule-8:- Create a separate clause corresponding to each conjunct. A WFF is true if all the clauses generated from it are true. The set of facts represented with WFF and set of clauses are equivalent.

In our example there is no AND operator thus it leads to only one clause.

$$\neg roman(x) \vee \neg know(x, Marcus) \vee$$
$$hate(x, Caesar) \vee \neg hate(y, z) \vee$$
$$thinkcrazy(x, y)$$

Converting a Fact into clause form.

Rule-9:- Standardize the variables such that each clause refers to different variable. Use following fact if necessary

$$(\forall x : P(x) \wedge Q(x)) \equiv \forall x : P(x) \wedge \forall y : Q(y)$$

In our case there is only one clause thus there is no need to apply this rule.

$$\neg roman(x) \vee \neg know(x, Marcus) \vee$$
$$hate(x, Caesar) \vee \neg hate(y, z) \vee$$
$$thinkcrazy(x, y)$$

Clause Form

Problem-1

- **Convert above fact into Clause form.**

$$\forall x : dog(x) \rightarrow hasatail(x) \wedge bark(x)$$

$Rule - 1$

$$\boxed{\forall x : \neg dog(x) \vee (hasatail(x) \wedge bark(x))}$$

$Rule - 2,3,4,$ *and* 5 *are not applicable*

$Rule - 6 -$ *drop the prefix*

$$\boxed{\neg dog(x) \vee (hasatail(x) \wedge bark(x))}$$

$Rule - 7 -$ *distribute* $\vee$ *over* $\wedge$

$$\boxed{(\neg dog(x) \vee hasatail(x)) \wedge (\neg dog(x) \vee bark(x))}$$

$Rule - 8 -$ *prepare seperate clauses*

$$\boxed{\begin{array}{l} 1. \neg dog(x) \vee hasatail(x) \\ 2. \neg dog(x) \vee bark(x) \end{array}}$$

$Rule - 9 -$ *change variable names*

$$\boxed{\begin{array}{l} 1. \neg dog(x) \vee hasatail(x) \\ 2. \neg dog(y) \vee bark(y) \end{array}}$$

Problem.2

Convert following WFF into clause form

$$\forall x : \exists y : drive(y) \wedge cook(y)$$

Rule 1 ,2 , 3 4 are not applicable

Rule 5- Eliminate Existential quantifiers by "Skolem functions"

$$\forall x : drive(s1) \wedge cook(s2)$$

S1 and s2 are "Skolem functions" and they correctly return value of x to satisfy the predicate.

Rule-6:- drop the prefix

$$drive(s1) \wedge cook(s2)$$

Rule-7-not applicable since it is already in CNF

Rule-8- creating separate clauses from each conjunct

$$\boxed{\begin{array}{l} 1.drive(s1) \\ 2.cook(s2) \end{array}}$$

Rule-9- Assuming that s1 will refer to x and s2 to y.

Resolution

- **It is a proof generating procedure.**
- **It generates proof by "Contradiction".**

Definitions.

1. **Contradiction:- is a statement whose every possible interpretation is "False" e.g**

$$(P \wedge \neg P)$$

2.Tautology:- is a statement whose every possible interpretation is "True" e.g

$$(P \vee \neg P)$$

3. Contingency:- is a statement which is neither tautology nor contradiction.

Resolution

- Resolution generates proof by "Refutation", means the fact to be proved is negated and then it is added to the knowledgebase and if a contradiction is found then original fact is proved.

~~blue(sky)~~ Red(carpet)	¬ ~~blue(sky)~~

Contradiction is found in two ways.

1. Null resolvant
2. A non-null resolvant which is a contradiction.

Unification

- In resolution we can resolve/cancel following clauses in a straight forward manner.

$$\left.\begin{array}{l} man(Marcus) \\ \neg man(Marcus) \end{array}\right\} \text{Parent Clauses.}$$

- But we can not resolve/cancel following clauses in a straight forward manner unless x is replaced by Marcus.

$$man(Marcus)$$

$$\neg man(x)$$

Unification

- Unification is a process that suggest substitution to unify the parent clauses.

Let L1 and L2 are parent clauses to be unified

Unification(L1, L2)

{ 1. If L1 = L2 return NULL

2. If L1 and L2 have same predicate symbol & same number of Arguments.

for each i th Argument in L1 and L2

substitution (A_{1i}/A_{2i})

if any one or both of them are Variables

else

report failure

}

Example of unification

$$hate(x, y)$$

$$\neg hate(Marcus, z)$$

$$\Downarrow$$

$$\left.\begin{array}{l} Marcus / x \\ y / z \end{array}\right\} \text{substitution}$$

Problem-3.

- Represent above facts in predicate logic and convert them into clause form.

Consider following facts about Santa Maria club.

1. Joe, Sally, Bill and Ellen are members of Santa Maria club.
2. Joe is married to sally.
3. Bill is Ellen's brother.
4. The spouse of every married person in the club is also in the club.
5. Last meeting of the club was at Joe's house.

Resolution

- Consider set of facts F and a statement to be proved P.

Resolution Algorithm

1. Convert all the facts in F to *clause form*.
2. *Negate P* and convert the result to clause form. Add it to the set of clauses obtained in step 1.
3. Repeat until either a contradiction is found, no progress can be made or a predetermined amount of effort has been expanded.
   a) Select parent clauses.
   b) Resolve them together. If necessary call to unification to unify them. The resolvant will be disjunction of all the literal from both the clauses. Omit the resolved literals from the resolvant.

Resolution

c) **If the resolvant is empty clause, then a contradiction has been found. If not then add it to the set of clauses available to the procedure. If can't progress with non-null resolvant, check that is it a contradiction?**

Guidelines to select clauses to speedup Resolution

- **Only resolve pairs of clauses that contain complementary literals.**
- **Eliminate certain clauses as soon as they are generated so that they can not participate in later process.**
- **Whenever possible resolve either with one of the clauses that is part of the statement to be proved or with a clause in resolvant. (*set of support strategy*)**
- **Wherever possible resolve with a clause having single literal. Such resolution generates resolvant with fewer literals.(*unit-preference strategy*)**

Example-1 of resolution.

1. $man(Marcus)$
2. $pompeian(Marcus)$
3. $\neg pompeain(x1) \vee roman(x1)$
4. $ruler(Caesar)$
5. $\neg roman(x2) \vee loyalto(x2, Caesar) \vee hate(x2, Caesar)$
6. $loyalto(x3, f1(x3))$
7. $\neg man(x4) \vee \neg ruler(y1) \vee \neg tryassassinate(x4, y1) \vee \neg loyalto(x4, y1)$
8. $tratassasinate(Marcus, Caesar)$

F

This is a set of facts about the Marcus in clause form. Suppose we want to prove that "Marcus hate Caesar".

$$hate(Marcus, Caesar)$$

P

Example-1 of resolution.

$\neg\ hate(Marcus, Caesar)$ **5**

Marcus/x2

3 $\neg roman(Marcus) \vee loyalto(Marcus, Caesar)$

Marcus/x1

2 $\neg pompeain(Marcus) \vee loyalto(Marcus, Caesar)$

7 $loyalto(Marcus, Caesar)$

Marcus/x4, Caesar/y1 **1**

$\neg man(Marcus) \vee \neg ruler(Caesar) \vee \neg tryassassinate(Marcus, Caesar)$

$\neg ruler(Caesar) \vee \neg tryassas\sin ate(Marcus, Caesar)$

$\neg ruler(Caesar) \vee \neg tryassassinate(Marcus, Caesar)$ **4**

8 $\neg tryassassinate(Marcus, Caesar)$

Null

Since we found a Null resolvant, it means that contradiction has been found and thus "Marcus hate Caesar" is proved.

Example-2 of resultion.

Consider following facts about Santa Maria club.
1. Joe, Sally, Bill and Ellen are members of Santa Maria club.
2. Joe is married to sally.
3. Bill is Ellen's brother.
4. The spouse of every married person in the club is also in the club.
5. Last meeting of the club was at Joe's house.

Knowledgebase of about this club in Clause Form is as follows.

1. $member(Joe)$
2. $member(Sally)$
3. $member(Bill)$
4. $member(Ellen)$
5. $married(Joe, Sally)$
6. $brother(Bill, Ellen)$
7. $\neg married(x, y) \vee \neg member(x) \vee member(y)$
8. $lastmeeting(Joe)$

Suppose we want to prove that "Ellen is not married"

$$\neg married(Ellen, z)$$

******************************Resolution*****************************

$married(Ellen, z)$ **7**

Ellen/x, z /y

4 $\neg member(Ellen) \vee member(z)$

$member(z)$

Z can assume following values z=Joe, z=Sally, z=Bill

$z \neq Joe$, *Since Joe is already married*

$z \neq Bill$, *Since Bill is Ellans brother*

$z \neq Sally$, *Since is already married with Joe*

Since every possible interpretation of member(z) is False, Thus it is a contradiction.

Problem.1

- **Represent following facts in predicate logic and using resolution prove that West is a criminal.**

  1. *It is crime for American to sell weapons to hostile nations.*

  2. Nono has some missiles.

  3. All of missiles owned by Nono were sold to it by Colonel West.

  4. Missiles are weapons.

  5. America counts enemy as hostile.

  6. West is American.

  7. Country Nono is enemy of America.

Reference:- Artificial intelligence A modern approach-Russell & Norvig, Page.280

Facts in WFF

$$American(x) \wedge weapon(y) \wedge sells(x, y, z) \wedge hostile(z) \Rightarrow cri\min al(x)$$
$$\exists x : owns(Nono, x) \wedge missile(x)$$
$$missile(x) \wedge owns(Nono, x) \Rightarrow sells(West, x, Nono)$$
$$missile(x) \Rightarrow weapon(x)$$
$$enemy(x, America) \Rightarrow hostile(x)$$
$$american(West)$$
$$enemy(Nono, America)$$

Facts in clause form

$\neg American(x) \vee \neg Weapon(y) \vee \neg Sells(x,y,z) \vee \neg Hostile(z) \vee Criminal(x)$

$American(West)$

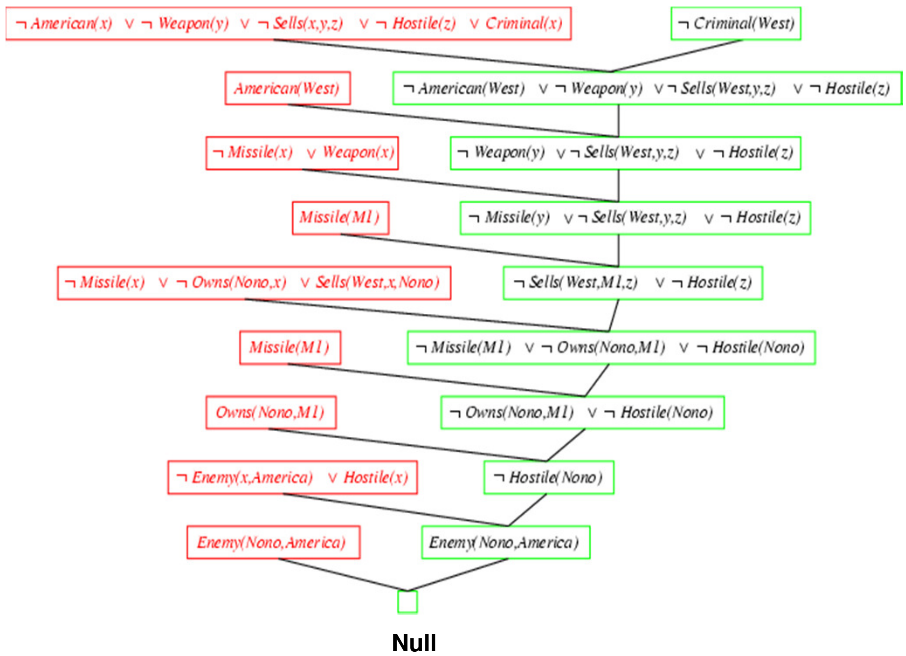
$\neg Missile(x) \vee Weapon(x)$

$Missile(M1)$

$Owns(Nono, M1)$

$\neg Enemy(x, America) \vee Hostile(x)$

$Enemy(Nono, America)$



Points to be learned

- Forward and Backward reasoning , Forward and Backward chaining rules.

***Reference*:- Rich & Knight.**

Introduction to Prolog

- The name **prolog** was taken from phrase "**PRO**gramming in **LOG**ic" and first developed in 1972, in France.
- It is unique in its ability to **infer facts** and **conclusions**
- User has to provide a **Knowledge Base (KB)** about a problem domain and has to specify a **goal**. Then system uses **KB** to achieve this **goal**, defining its own procedure.
- **Procedural languages:-**
  - like C, C++, uses procedure describing how to solve a given problem?.
  - Focus on Procedure and Data
  - Same procedure is executed many times with great speed

Introduction to Prolog

- **Prolog Features**
  - OOP language
  - No procedure and thus, no programs and thus no programmers, but "Knowledge Engineers"
  - Only data (facts) about objects and their relationships
  - Thus, a prolog program= database (Knowledge base)
  - ***Goal*** is given by user and using formal reasoning it proves or disproves it for find truth value
  - It is a complied language

Introduction to Prolog

- **Prolog software : This course recommends SWI prolog**
- **SWI Prolog available on**
- [1]. http://www.swi-prolog.org/download/stable
- **Tutorials on**
- [2]. http://lpn.swi-prolog.org/lpnpage.php?pagetype=html&pageid=lpn-htmlse1
- **Learn:-** Case study of Expert system from Rich and Knight (MYCIN OR PROSPECTOR)
- Following are corresponding **Lab Assignments**

Lab assignment 4:- Implement real time applications in Prolog.
Uses Knowledge bases KB3 to KB5 and all the queries on them in [2]
Lab assignment 5:- Expert System in Prolog
Implement a small expert system in prolog for detection of childhood diseses
(refer:- Introduction to Turbo Prolog by Carl Townsend)