Mathematical Foundations of Computing

Predicate Calculus : 2

N Geetha

AM & CS

PSG College of Technology

Normal Forms in Predicate Calculus

- A FOL expression Φ can be put in **prenex normal form** by moving the quantifiers to the prefix. The quantifier free part of Φ is called the matrix of Φ

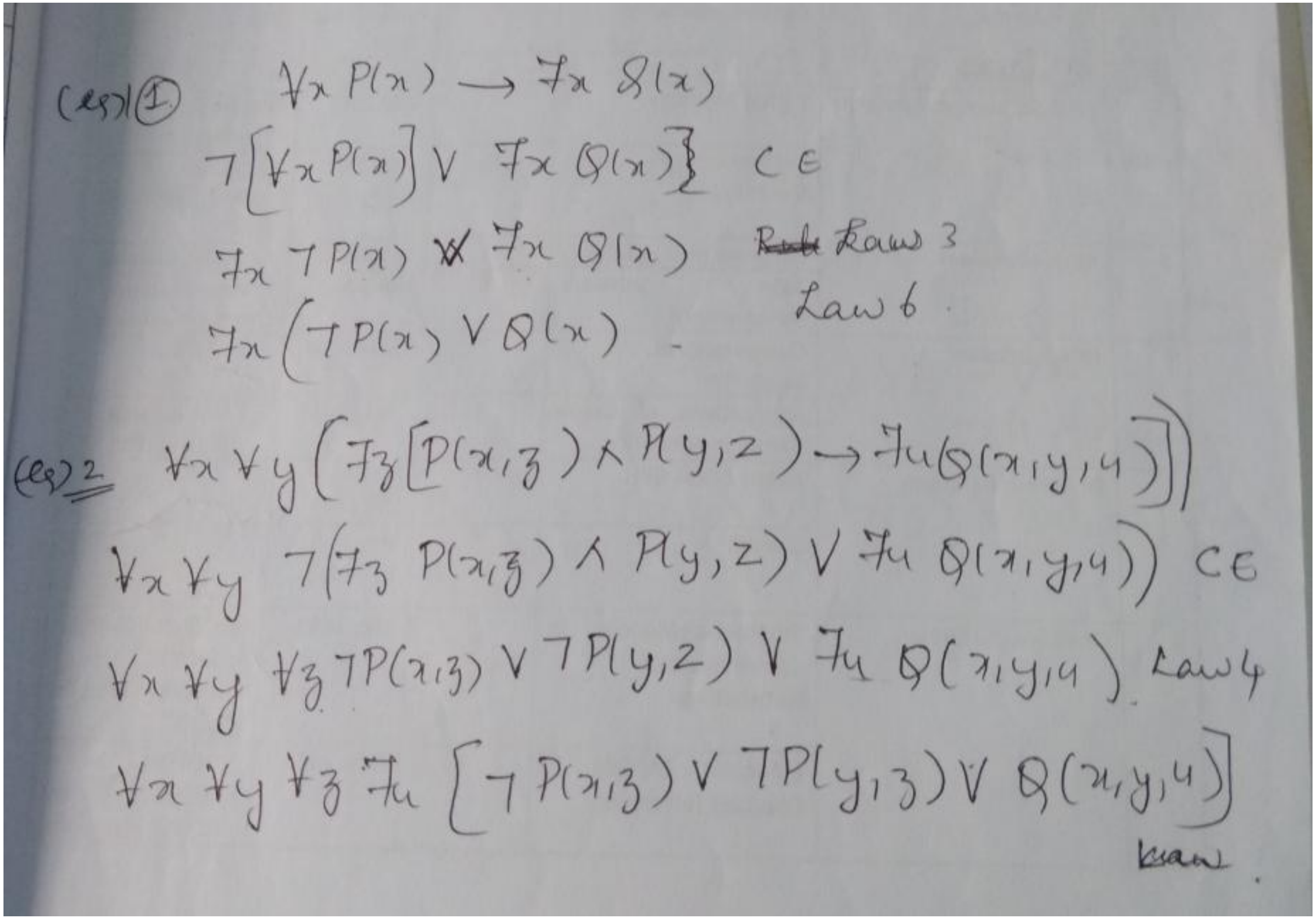
  ϕ: Q[1]V[1].$\cdots$ .Q[n]V[n]. *Quantifier-free formula*

  where Q[i] $\in \{\forall, \exists\}$ and V[i] is a variable.
- Every formula has a logically equivalent Prenex Normal Form
- Eg. $(\forall x)(\forall y)[P(x,y) \land Q(y)]$
- $(\forall x)(\exists z)[\neg P(x,z) \rightarrow Q(z)]$
- Exer: 1. $\forall x\, P(x) \land \forall x\, Q(x)$
- 2. $(\forall x)(\forall y)\neg[P(x) \rightarrow Q(y)]$
- 3. $\forall x \exists y\, R(x,y)$
- 4. $R(x,y)$
- 5. $\neg \forall x\, R(x,y)$

Working Rule to tranform a formula into Prenex Normal Form

1. Eliminate $\rightarrow$ and $\leftrightarrow$ (transform to $\vee$, $\wedge$, $\neg$)
2. Push negations inside using Involution, DeMorgan's and the laws
   a) $\neg (\forall x\, p(x)) \Leftrightarrow \exists x(\neg\, p(x))$
   b) $\neg (\exists x\, p(x)) \Leftrightarrow \forall x(\neg\, p(x))$
3. If there are name conflicts across scopes, solve with renaming.
4. Use the equivalence laws (1 to 8) to move the quantifiers to the left of the entire formula to ultimately obtain in Prenex Normal Form

PNF Examples



(eg) 1 $\forall x\, P(x) \rightarrow \exists x\, Q(x)$

$\neg[\forall x\, P(x)] \vee \exists x\, Q(x)$ CE

$\exists x\, \neg P(x) \vee \exists x\, Q(x)$ ~~Rule~~ Law 3

Law 6

$\exists x(\neg P(x) \vee Q(x))$.

(eg) 2 $\forall x \forall y (\exists z [P(x,z) \wedge P(y,z) \rightarrow \exists u\, Q(x,y,u)])$

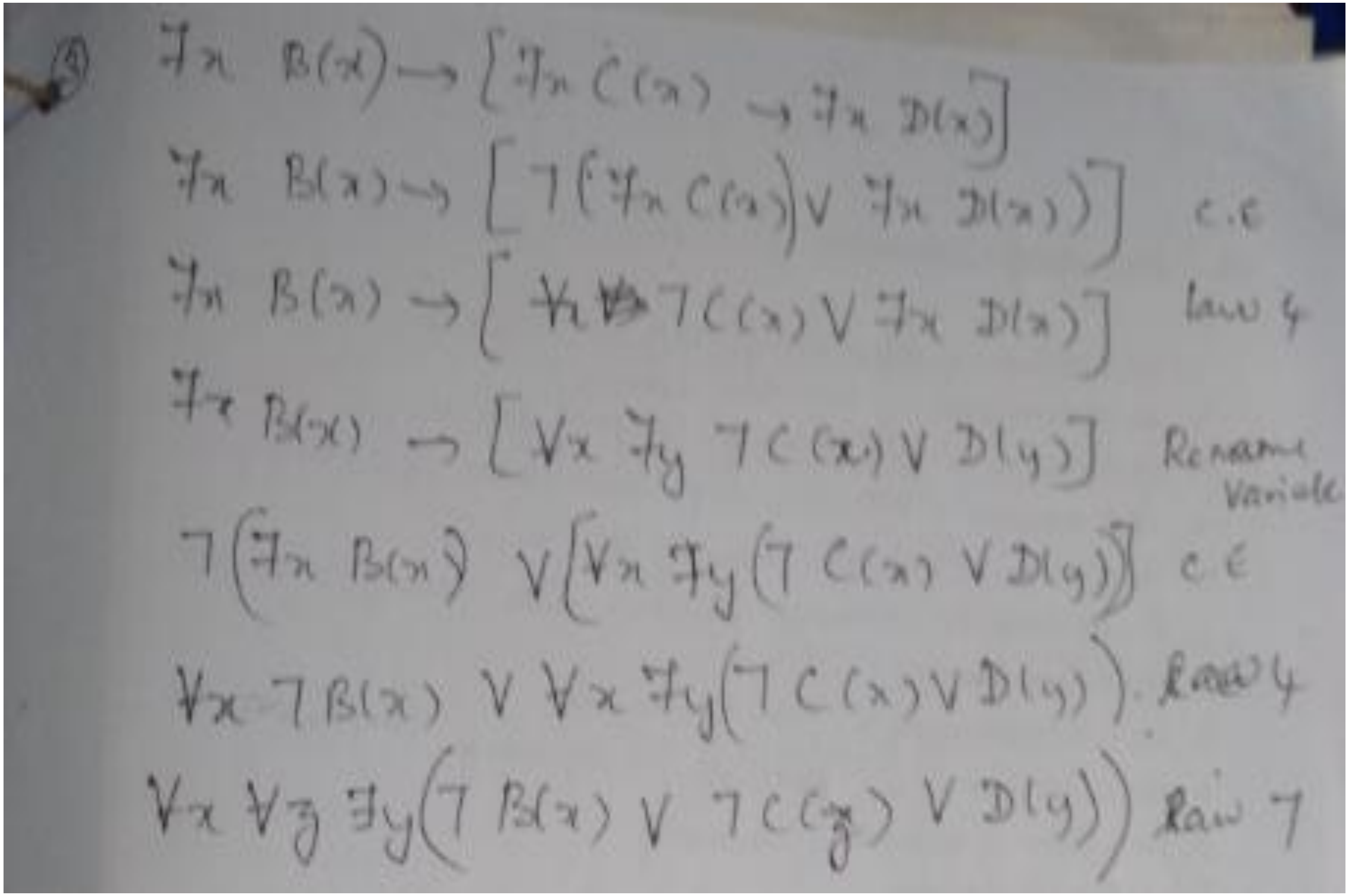
$\forall x \forall y\ \neg(\exists z\, P(x,z) \wedge P(y,z)) \vee \exists u\, Q(x,y,u))$ CE

$\forall x \forall y \forall z\, \neg P(x,z) \vee \neg P(y,z) \vee \exists u\, Q(x,y,u)$ Law 4

$\forall x \forall y \forall z \exists u\, [\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,u)]$

Law .

PNF Examples



(3) $\exists x\, B(x) \rightarrow [\exists x\, C(x) \rightarrow \exists x\, D(x)]$

$\exists x\, B(x) \rightarrow [\neg(\exists x\, C(x)) \vee \exists x\, D(x))]$ c.e

$\exists x\, B(x) \rightarrow [\forall x\, \neg C(x) \vee \exists x\, D(x)]$ law 4

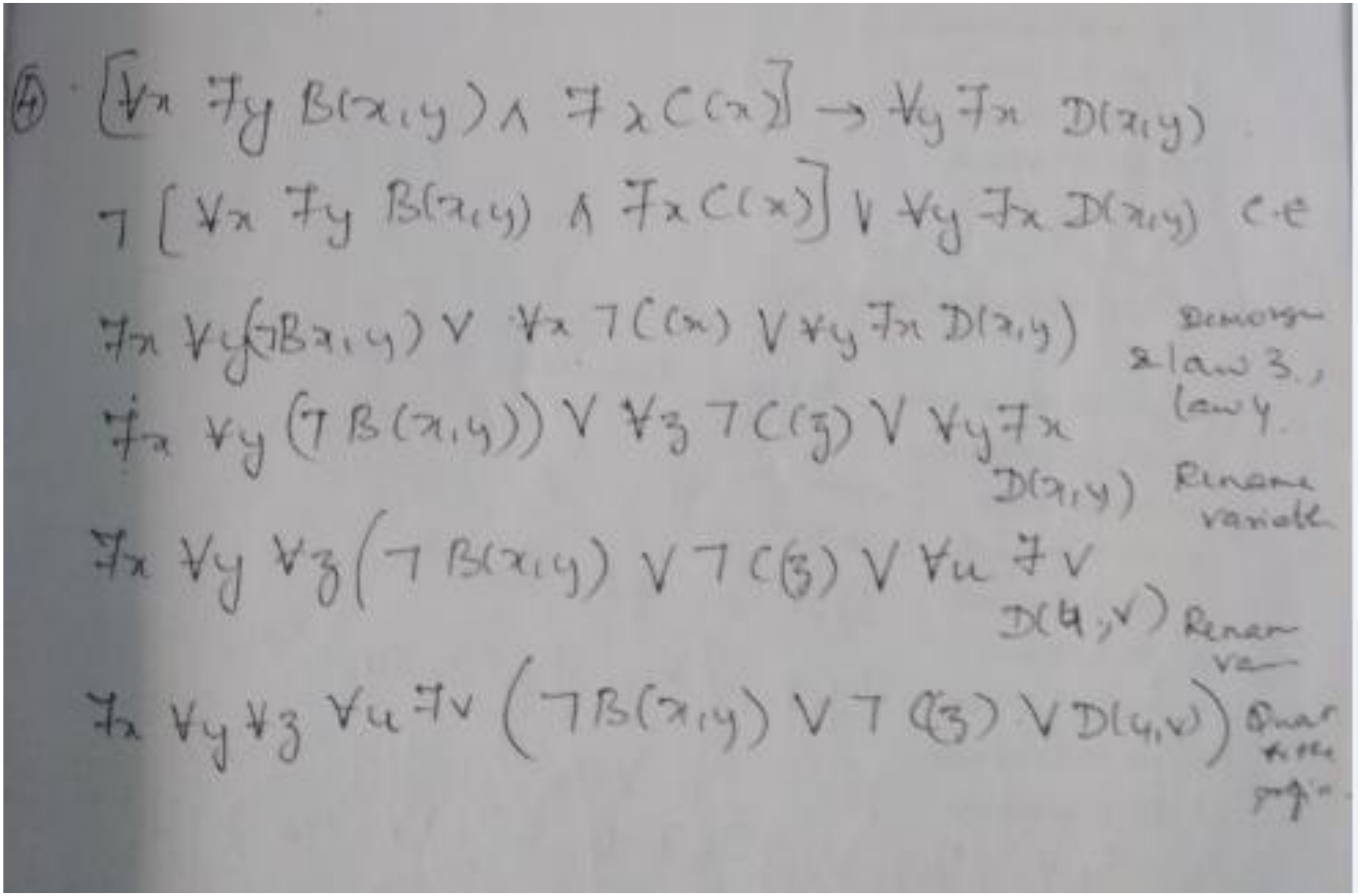
$\exists x\, B(x) \rightarrow [\forall x\, \exists y\, \neg C(x) \vee D(y)]$ Rename Variable

$\neg(\exists x\, B(x)) \vee [\forall x\, \exists y (\neg C(x) \vee D(y))]$ c.e

$\forall x\, \neg B(x) \vee \forall x\, \exists y (\neg C(x) \vee D(y))$. Law 4

$\forall x\, \forall z\, \exists y (\neg B(x) \vee \neg C(z) \vee D(y))$ Law 7

PNF Examples



④. $[\forall x\, \exists y\, B(x,y) \wedge \exists x\, C(x)] \rightarrow \forall y\, \exists x\, D(x,y)$

$\neg[\forall x\, \exists y\, B(x,y) \wedge \exists x\, C(x)] \vee \forall y\, \exists x\, D(x,y)$ c.e

$\exists x\, \forall y(\neg B(x,y)) \vee \forall x\, \neg C(x) \vee \forall y\, \exists x\, D(x,y)$ Demorgan & law 3., law 4.

$\exists x\, \forall y\, (\neg B(x,y)) \vee \forall z\, \neg C(z) \vee \forall y\, \exists x\, D(x,y)$ Rename variable

$\exists x\, \forall y\, \forall z\, (\neg B(x,y) \vee \neg C(z) \vee \forall u\, \exists v\, D(u,v))$ Rename var

$\exists x\, \forall y\, \forall z\, \forall u\, \exists v\, (\neg B(x,y) \vee \neg C(z) \vee D(u,v))$ Quar to the [illegible]

Find PNF

- 5. $\forall x\, [\forall y\, A(y,x) \rightarrow \exists y\, B(x,y)\,]$
- 6. $\exists z\, [\exists x\, Q(x,z) \vee \exists x\, P(x)] \rightarrow \neg\, [\neg\, \exists x\, P(x) \wedge \forall x\, \exists z\, Q(z,x)]$
- 7. $\neg\, \exists y\, [\forall x\, P(x) \rightarrow \forall x\, Q(x,y)]$
- 8. $\exists z(\exists x Q(x, z) \vee \exists x P(x)) \rightarrow \neg(\neg\exists x P(x) \wedge \forall x\exists z\, Q(z, x))$.
- PNF is required to
  - Simplify the structure
  - Proceed to Skolem Normal Form
  - Do Automated Theorem Proving

Skolem Normal Form

- **Skolemization:** procedure for systematic elimination of the existential quantifiers in a first-order formula in a prenex form, by introducing new constant and functional symbols, called Skolem constants and Skolem functions, in the formula and the matrix should be in CNF.
- $Q_1(x_1)Q_2(x_2)..Q_n(x_n)$ (F) F : Matrix in CNF
- Rule 1: Check that no universal quantifier appear before Q_r (Q_r is an existential quantifier). Then choose a new constant different from other constants occurring in F and replace all x_r appearing in the F by the constant
- Eg1. $\exists x \forall y \forall z\ [A(x,y,z) \wedge Q(x)]$ is the formula
-   $x=c$; $\forall y \forall z\ [A(c,y,z) \wedge Q(c)]$, c is a new (Skolem) constant.
- Eg2. $\exists x \forall y \forall z(P(x, y) \rightarrow Q(x, z))$ is $\forall y \forall z(P(c, y) \rightarrow Q(c, z))$.
- **Note that the resulting formula is not equivalent to the original one, but is equally satisfiable with it.**

Skolem Normal Form

- Rule 2: If for existential quantifier Q_r (the quantifiers Q_{s1} Q_{s2} Q_{s3} .. Q_{sm} are all universal quantifiers appearing before Q_r), then we choose a function f, different from other function symbols occurring in F and replace all x_r in F by the function $f(x_{s1},x_{s2},\ldots x_{sm})$ and delete $Q_r x_r$ from the prefix
- Eg1. $\forall x\forall y\exists z\,[A(x,y,z) \wedge Q(y,z)]$ is the formula
- $z=f(x,y)$; $\forall x\forall y\,[A(x,y,f(x,y)) \wedge Q(y,f(x,y))]$, f is a new (Skolem) function.
- Eg2. $\exists x\ \forall y\ \forall z\ \exists u\ \forall v\ \exists w\ [A(x,y,z,u,v,w)]$?
- Eg3. $\forall x\ \exists y\ \forall z\ [(\neg P(x,y) \wedge Q(x,z)) \vee R(x,y,z)]$?
- Eg4. If everybody likes somebody, then that person is a king.
- **Note that the resulting formula is not equivalent to the original one, but is equally satisfiable with it.**

Skolemization

- Skolem provided a technique to convert a FOL sentence F into F’ in prenex normal form with universal quantifiers such that F is satisfiable iff F’ is satisfiable
- FOL with resolution is refutation complete, i.e. if S is a set of unsatisfiable clauses, then a contradiction arises after a finite number of resolutions.

Unification

- ***Unification***: The process of finding all legal substitutions that make logical expressions look identical
- This is a recursive algorithm
- Literals may match exactly as they stand eg. King(John) and King(John)
- Literals can be made to match by an appropriate substitution
  - Can replace a variable by a constant :
    King(x) and King(John)   x/John
  - Can replace a variable by a variable :
    King(x) and King(y)   x/y
  - Can replace a variable by a function expression :
    King(x) and King(father(Richard))   x/father(Richard))
- A variable cannot be unified with a term containing that variable. The test for it is called the occurs check.
  - e.g., X cannot unify with F(X, b)

Unification

- If p and q are logical expressions, then Unify(p,q) = Substitution List S if using S makes p and q identical or fail.
- Can find a substitution θ such that King(x) and Greedy(y) match King(John) and Greedy(John)
- $\theta = \{x/John, y/John\}$ works
- Unify(α,β) = θ if $\alpha\,\theta = \beta\,\theta$
- Standardize apart: before unifying, make sure that p and q contain different variable names.

function unify code

```
function unify(E1, E2);
   begin
      case
         both E1 and E2 are constants or the empty list:                    %recursion stops
            if E1 = E2 then return {}
               else return FAIL;
         E1 is a variable:
            if E1 occurs in E2 then return FAIL
               else return {E2/E1};
         E2 is a variable:
            if E2 occurs in E1 then return FAIL
               else return {E1/E2}
         either E1 or E2 are empty then return FAIL        %the lists are of different sizes
         otherwise:                                         %both E1 and E2 are lists
            begin
                  HE1 := first element of E1;
                  HE2 := first element of E2;
                  SUBS1 := unify(HE1,HE2);
                  if SUBS1 : = FAIL then return FAIL;
                  TE1 := apply(SUBS1, rest of E1);
                  TE2 : = apply (SUBS1, rest of E2);
                  SUBS2 : = unify(TE1, TE2);
                  if SUBS2 = FAIL then return FAIL;
                        else return composition(SUBS1,SUBS2)
            end
   end                                                                   %end case
end
```

Unification

p	q	θ
$Knows(John, x)$	$Knows(John, Jane)$	$\{x/Jane\}$
$Knows(John, x)$	$Knows(y, OJ)$	$\{x/OJ, y/John\}$
$Knows(John, x)$	$Knows(y, Mother(y))$	$\{y/John, x/Mother(John)\}$
$Knows(John, x)$	$Knows(x, OJ)$	$fail$

Resolution Theorem Proving (FOL)

- 1. Convert Hypothesis to Clause form
- 2. Convert conclusion to clause form
- 3. Start with Neg(Conclusion) as parent1 and choose parent2
- 4. Resolve using unification; If an empty resolvent is found, then return success.
- 5. If not possible return failure.
- 6. Use the resolvent as a parent and continue the process.

- If resolution is successful, proof succeeds
- If there was a variable in the item to prove, return variable's value from unification bindings

Eg1.

Facts :

1. $\neg P(x) \lor P(f(x))$
2. $\neg Q(a, y) \lor \neg R(y, x) \lor P(x)$
3. $R(b, g(a, z))$
4. $Q(a, b)$

Goal: $P(f(g(a, c)))$

- Use resolution to prove goal.

Example 2

- The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Col. West, who is an American.

- Prove that Col. West is a criminal.

Resolution in Predicate Calculus Eg

1. It is a crime for an american to sell weapons to a hostile country.

1’. $\forall x\ \forall y\ \forall z[\text{American}(x) \wedge \text{Weapon}(y) \wedge \text{Hostile}(z) \wedge \text{Sell}(x,y,z) \rightarrow \text{Criminal}(x)]$

2. The country Nono has some missiles.

   $\exists x\ \text{Owns}(\text{Nono},x) \wedge \text{Missile}(x)$.

2’. Missile(M1). … Skolem Constant introduction

2’’. Owns(Nono,M1).

Prove: West is a criminal

3. All of its missiles where sold to it by Colonel West.

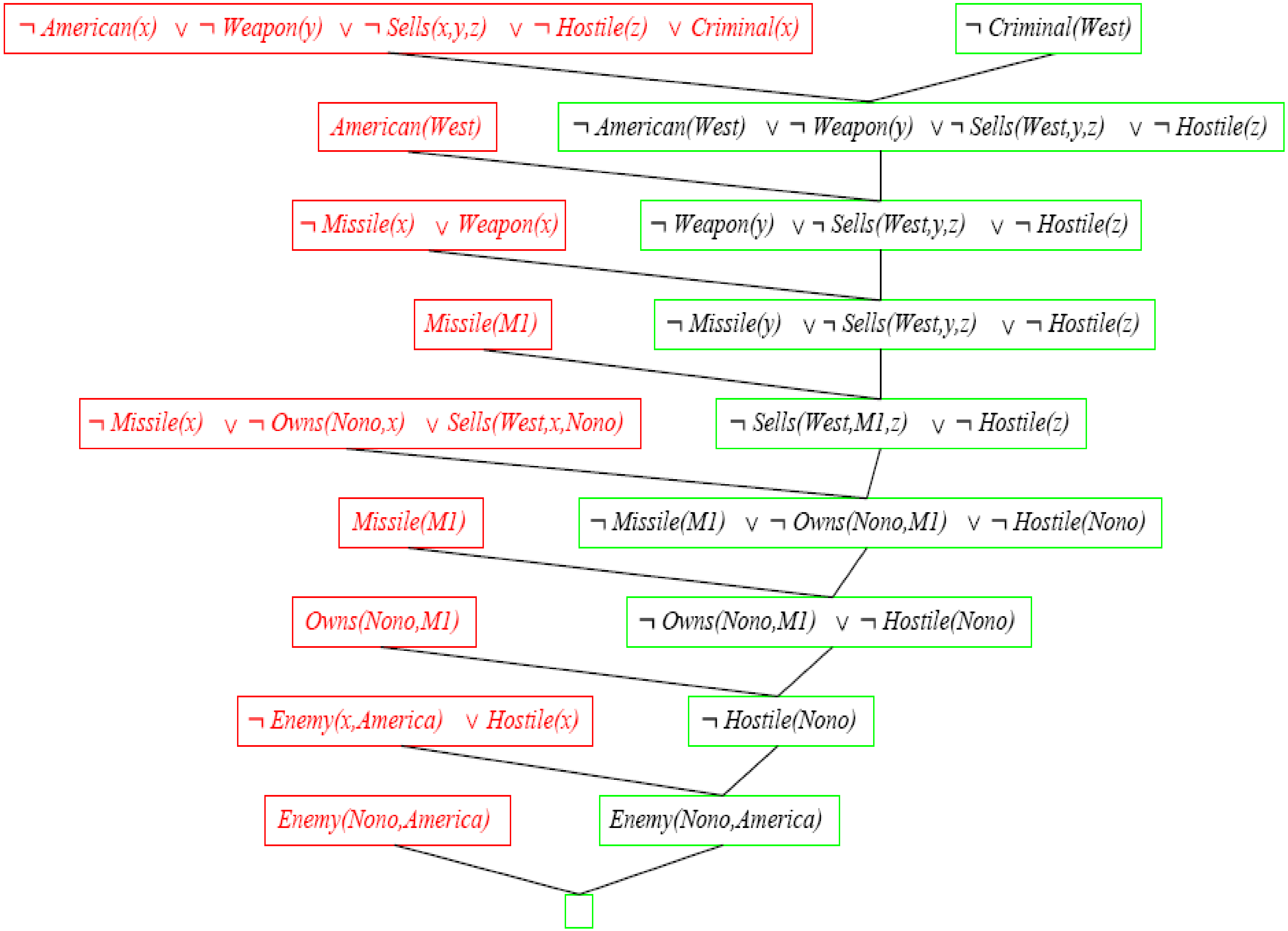
3'. $\forall x$ [Missile(x)^Owns(Nono,x) $\rightarrow$ Sells(West,x,Nono)]

4'. $\forall x$ [Missile(x) $\rightarrow$ Weapon(x)] "common sense"

5'. $\forall x$ Enemy(x, America) $\rightarrow$ Hostile(x)]

6'. American(West)

7'. Enemy(Nono, America).


¬ American(x) ∨ ¬ Weapon(y) ∨ ¬ Sells(x,y,z) ∨ ¬ Hostile(z) ∨ Criminal(x)
¬ Criminal(West)
American(West)
¬ American(West) ∨ ¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ Weapon(x)
¬ Weapon(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(y) ∨ ¬ Sells(West,y,z) ∨ ¬ Hostile(z)
¬ Missile(x) ∨ ¬ Owns(Nono,x) ∨ Sells(West,x,Nono)
¬ Sells(West,M1,z) ∨ ¬ Hostile(z)
Missile(M1)
¬ Missile(M1) ∨ ¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
Owns(Nono,M1)
¬ Owns(Nono,M1) ∨ ¬ Hostile(Nono)
¬ Enemy(x,America) ∨ Hostile(x)
¬ Hostile(Nono)
Enemy(Nono,America)
Enemy(Nono,America)

Predicate Calculus: Eg3.

- A, B and C belong to the Himalayan club. Every member in the club is either a mountain climber or a skier or both. A likes whatever B dislikes and dislikes whatever B likes. A likes rain and snow. No mountain climber likes rain. Every skier likes snow.
- *Is there a member who is a mountain climber and not a skier?*
- Given knowledge has:
  - Facts
  - Rules

Predicate Calculus: Example

- Let *mc* denote mountain climber and *sk* denotes skier. Knowledge representation in the given problem is as follows:
  1. *member(A, Hclub)*
  2. *member(B, Hclub)*
  3. *member(C, Hclub)*
  4. $\forall x[member(x,Hclub) \rightarrow (mc(x) \vee sk(x))]$
  5. $\forall x[mc(x) \rightarrow \sim like(x,rain)]$
  6. $\forall x[sk(x) \rightarrow like(x, snow)]$
  7. $\forall x[like(B, x) \rightarrow \sim like(A, x)]$
  8. $\forall x[\sim like(B, x) \rightarrow like(A, x)]$
  9. *like(A, rain)*
  10. *like(A, snow)*
  11. *Question:* $\exists x[member(x,Hclub) \wedge mc(x) \wedge \sim sk(x)]$
- We have to infer the 11th expression from the given 10.
- Use Resolution Refutation.

Club example: Inferencing

1. $member(A, Hclub)$
2. $member(B, Hclub)$
3. $member(C, Hclub)$
4. $\forall x[member(x, Hc\text{lub}) \rightarrow (mc(x) \vee sk(x))]$
   - Can be written as $[member(x, Hc\text{lub}) \rightarrow (mc(x) \vee sk(x))]$
   - $\sim member(x, Hc\text{lub}) \vee mc(x) \vee sk(x)$
5. $\forall x[sk(x) \rightarrow lk(x, snow)]$
   - $\sim sk(x) \vee lk(x, snow)$
6. $\forall x[mc(x) \rightarrow \sim lk(x, rain)]$
   - $\sim mc(x) \vee \sim lk(x, rain)$
7. $\forall x[like(A, x) \rightarrow \sim lk(B, x)]$
   - $\sim like(A, x) \vee \sim lk(B, x)$

8. $\forall x[\sim lk(A,x) \rightarrow lk(B,x)]$

– $lk(A,x) \vee lk(B,x)$

9. $lk(A,rain)$

10. $lk(A,snow)$

11. $\exists x[member(x,Hc\text{lub}) \wedge mc(x) \wedge \sim sk(x)]$

– Negate–

$$\forall x[\sim member(x,Hc\text{lub}) \vee \sim mc(x) \vee sk(x)]$$

- Now standardize the variables apart which results in the following

1. *member(A, Hclub)*
2. *member(B, Hclub)*
3. *member(C), Hclub*
4. $\sim member(x_{1,} Hc\text{lub}) \vee mc(x_1) \vee sk(x_1)$
5. $\sim sk(x_2) \vee lk(x_2, snow)$
6. $\sim mc(x_3) \vee \sim lk(x_3, rain)$
7. $\sim like(A, x_4) \vee \sim lk(B, x_4)$
8. $lk(A, x_5) \vee lk(B, x_5)$
9. $lk(A, rain)$
10. $lk(A, snow)$
11. $[\sim member(x_{6,} Hc\text{lub}) \vee \sim mc(x_6) \vee sk(x_6)]$

Eg3.

- Now apply resolution and complete the problem

4. Prove using Resolution

1. John likes all kinds of food
2. Apples are food.
3. Chicken is food.
4. Anything that anyone eats and isn’t killed by is food.
5. Bill eats peanuts and is still alive.
6. Sue eats everything Bill eats.
7. Prove that John likes peanuts.
8. What food does Sue eat ?

5. Prove using Resolution

- Horses are faster than dogs and there is a greyhound that is faster than every rabbit. We know that Harry is a horse and that Ralph is a rabbit. Derive that Harry is faster than Ralph.

- Horse(x) Greyhound(y)
- Dog(y) Rabbit(z)
- Faster(y,z)

$$\forall x\, \forall y\ \mathbf{Horse}(x) \land \mathbf{Dog}(y) \rightarrow \mathbf{Faster}(x,y)$$

$$\exists y\ \mathbf{Greyhound}(y) \land (\forall z\ \mathbf{Rabbit}(z) \rightarrow \mathbf{Faster}(y,z))$$

Horse(Harry)

Rabbit(Ralph)

$$\forall y\ \mathbf{Greyhound}(y) \rightarrow \mathbf{Dog}(y)$$

$$\forall x\, \forall y\, \forall z\ \mathbf{Faster}(x,y) \land \mathbf{Faster}(y,z) \rightarrow \mathbf{Faster}(x,z)$$

H1. $\forall x \ \forall y$ Horse(x) $\wedge$ Dog(y) $\rightarrow$ Faster(x,y)

H2. $\exists y$ Greyhound(y) $\wedge$ ($\forall z$ Rabbit(z) $\rightarrow$ Faster(y,z))

H3. Horse(Harry)

H4. Rabbit(Ralph)

H5. $\forall y$ Greyhound(y) $\rightarrow$ Dog(y)

H6. $\forall x \ \forall y \ \forall z$ Faster(x,y) $\wedge$ Faster(y,z) $\rightarrow$ Faster(x,z)

T Faster(Harry,Ralph)

C1. ~Horse(x) $\vee$ ~Dog(y) $\vee$ Faster(x,y)

C2. Greyhound(Greg)

C2' ~Rabbit(z) $\vee$ Faster(Greg,z)

C3. Horse(Harry)

C4. Rabbit(Ralph)

C5. ~Greyhound(y) $\vee$ Dog(y)

C6. ~Faster(x,y) $\vee$ ~Faster(y,z) $\vee$ Faster(x,z)

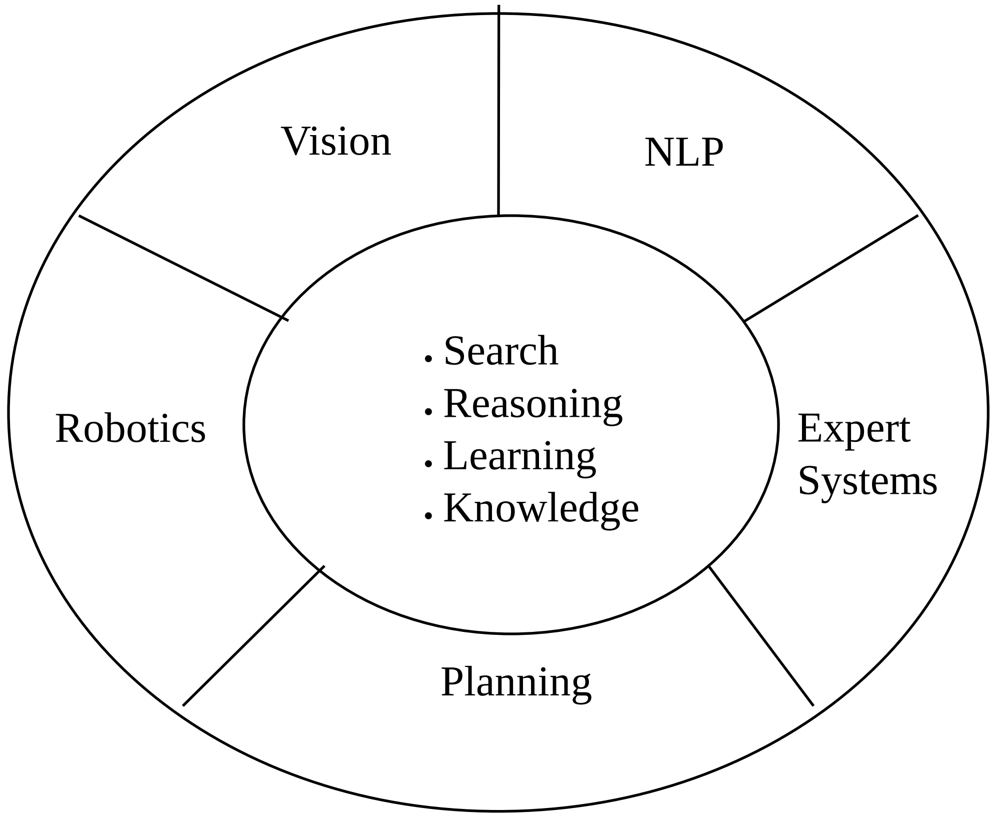
C7. ~Faster(Harry,Ralph)

Logic Programming

- Logic Programming
  - Identify problem
  - Assemble information
  - Encode information in KB
  - Encode problem instance as facts
  - Ask queries
  - Find false facts
- Lisp, Prolog are languages for logic

- Ordinary Programming
  - Identify problem
  - Assemble information
  - Figure out solution
  - Program Solution
  - Encode problem instance as data
  - Apply program to data
  - Debug procedural errors
- C++, Java, C#, etc

Logic and inferencing are useful in



Obtaining implication of given facts and rules -- Hallmark of intelligence

Summary

- Predicate calculus: allows quantified variables as parameters of predicates or functions
- Normal forms in Predicate Calculus
- Unification and Resolution in Predicate Calculus

References

1. Artificial Intelligence, by Elaine Rich and Kevin Knight
2. Artificial Intelligence Course, IIT Bombay