Artificial Intelligence

Logic Concepts

B.Tech. V Semester CSE

By
Dr. Loshma Gunisetti,
Associate Professor
CSE Department

Course Outcomes

After successful completion of this course, the student will be able to:

CO	Course Outcomes	Knowledge Level
1	Illustrate the concept of Intelligent systems and current trends in AI	K2
2	Apply Problem solving, Problem reduction and Game Playing techniques in AI	K3
3	**Illustrate the Logic concepts in AI**	**K2**
4	Explain the Knowledge Representation techniques in AI	K2
5	Describe Expert Systems and their applications	K2
6	Illustrate Uncertainty Measures	K2

Introduction

- Initially, Logic was considered to be a branch of philosophy, however, since the middle of the nineteenth century, formal logic has been studied in the context of foundations of mathematics, where it is often referred to as symbolic logic
- Logic helps in investigating and classifying the structure of statements and arguments through the study of formal systems of inference
- It is a study of methods that help in distinguishing correct reasoning from incorrect reasoning
- Formally, Logic is concerned with the principles of drawing valid inferences from a given set of true statements

Introduction

- Formal Logic deals with the study of inference with purely formal content
- It is often used as a synonym for Symbolic Logic, which is a study of symbolic abstractions
- Symbolic Logic is divided in two branches:
    1. Propositional Logic
    2. Predicate Logic
- In the study of Logic, a Proposition refers to a declarative statement that is either true or false in a given context
- A new proposition can be inferred from a given set of propositions in the same context using Logic
- An extension to symbolic logic is mathematical logic, concerned with study of proof theory, set theory, model theory and recursion theory .

Introduction

- Logical systems should possess properties such as :
  1) Consistency : None of the theorems of the system should contradict each other
  2) Soundness : The inference rules shall never allow a false inference from true premises
  3) Completeness : There are no true sentences in the system that cannot be proved in the system

Propositional Calculus

- Propositional Calculus (PC) refers to a language of propositions in which a set of rules are used to combine simple propositions to form compound propositions with the help of certain logical operators.
- The logical operators are called Connectives viz. not(~) , and (Λ), or (V) , implies (→), and equivalence(↔)
- In PC, concept of well formed formula is very important
- A well formed formula is defined as a symbol or a string of symbols generated by the formal grammar of a formal language
- Properties of a well formed formula in PC are:
  - ➢ The smallest unit (or an atom) is considered to be a well formed formula
  - ➢ If α is a well formed formula, then ~α is also a well formed formula
  - ➢ If α and β are well formed formulae, then (α Λ β), (α V β), (α → β) and (α↔β) are also well formed formulae
- A propositional expression is called a well-formed formula if and only if it satisfies the above properties

Truth Table

- In PC, a truth table is used to provide operational definitions of important logical operators; it elaborates all possible truth values of a formula
- The logical constants in PC are True (T) and False (F)
- Let us assume A, B, C are propositioned symbols

A	B	~A	A Λ B	A V B	A →B	A↔B
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Truth Table for Logical Operators

Truth Table

- The truth values of well formed formulae are calculated the truth table approach.

  For eg.

- Compute the truth value of α : (A V B) Λ (~B → A) using the truth table approach

A	B	A V B	~ B	~B →A	α
T	T	T	F	T	T
T	F	T	T	T	T
F	T	T	F	T	T
F	F	F	T	F	F

Truth Table for α

- Two formulae α and β are said to be logically equivalent ($\alpha \cong \beta$) if and only if the truth values of both are the same for all possible assignments of logical constants (T or F) to the symbols appearing in the formulae

Equivalence Laws

Equivalence relations (or laws) are used to reduce or simplify a given well-formed formula or to derive a new formula from the existing formula.

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

Propositional Logic

- Propositional Logic (or prop logic) deals with the validity , satisfiability(consistency) and unsatisfiability (inconsistency) of a formula and the derivation of a new formula using equivalence laws.
- Each row of a truth table for a given formula α is called its interpretation under which the value of a formula may be either true or false.
- A formula α is said to be a tautology if and only if the value of α is true for all its interpretations.
- A formula α is said to be **valid** if and only if it is a tautology
- A formula α is said to be **satisfiable** if there exists at least one interpretation for which α is true
- A formula α is said to be **unsatisfiable** if the value of α is false under all interpretations

Propositional Logic

- Example- Show that the following is valid argument:

If it is humid then it will rain and since it is humid today it will rain

Solution: Assume A: It is humid

B : It will rain

Then $\alpha : [(A \rightarrow B) \ \Lambda \ A] \rightarrow B$

Using truth table approach, we can see that α is true under all interpretations and hence is a **valid argument.**

Truth Table for $[(A \rightarrow B) \ \Lambda \ A] \rightarrow B$

A	B	A → B = (X)	X Λ A=(Y)	Y →B
T	T	T	T	T
T	F	F	F	T
F	T	T	F	T
F	F	F	F	T

Propositional Logic

- The truth table approach is a simple and straightforward method and is extremely useful at presenting an overview of all the truth values in a given situation
- Limitation of a truth table is that its size grows exponentially i.e. if a formula contains n atoms, then the truth table will contain 2^n entries
- In some cases, all entries of a truth table are not required. Construction of a truth table becomes a futile exercise
- Other alternatives to truth table are:
  - **Natural Deduction System**
  - **Axiomatic System**
  - **Semantic Tableau method**
  - **Resolution Refutation method**

Natural Deduction System (NDS)

- It mimics the pattern of natural reasoning
- It is based on a set of deductive inference rules
- Assuming that $A_1, A_2, \ldots, A_k$, $1 \le k \le n$, are a set of items and α_j , where $1 \le j \le m$, and β are well formed formulae, the inference rules may be stated as shown in the following NDS rules table

Rule Name	Symbol	Rule
Introducing Λ	(I : Λ)	If $A_1, \ldots, A_n$ then $A_1 \Lambda \ldots \Lambda A_n$
Eliminating Λ	(E : Λ)	If $A_1 \Lambda \ldots \Lambda A_n$ then A_i $(1 \le i \le n)$
Introducing V	(I : V)	If any A_i $(1 \le i \le n)$ then $A_1 V \ldots V A_n$
Eliminating V	(E : V)	If $A_1 V \ldots V A_{n,}$ $A_1 \rightarrow A, \ldots A_n \rightarrow A$ then A

NDS Rules Table

Natural Deduction System (NDS)

Rule Name	Symbol	Rule
Introducing →	(I : →)	If from $A_1, \ldots, A_n$ infer B is proved then then $A_1 \wedge \ldots \wedge A_n \rightarrow B$ is proved
Eliminating →	(E : →)	If $A_1 \rightarrow A$, A_1 , then A *(Modus Ponen Rule)*
Introducing ↔	(I : ↔)	If $A_1 \rightarrow A_2$, $A_2 \rightarrow A_1$ then $A_1 \leftrightarrow A_2$
Eliminating ↔	(E : ↔)	If $A_1 \leftrightarrow A_2$ then $A_1 \rightarrow A_2$, $A_2 \rightarrow A_1$
Introducing ~	(I : ~)	If from A infer $A_1 \wedge \sim A_1$ is proved then ~A is proved
Eliminating ~	(E : ~)	If from ~A infer $A_1 \wedge \sim A_1$ is proved then A is proved

NDS Rules Table (Contd.)

Natural Deduction System (NDS)

Prove that : A Λ (B V C) is deduced from A Λ B

Solution: The theorem in NDS can be written as **from** A Λ B **infer** A Λ (B V C)

Description	Formula	Comments
Theorem	from A Λ B infer A Λ (B V C)	To be proved
Hypothesis(given)	A Λ B	1
E : Λ (1)	A	2
E : Λ (1)	B	3
I: V (3)	B V C	4
I : Λ (2,4)	A Λ (B V C)	Proved

Proof of the Theorem

Deduction Theorem: To prove a formula $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \beta$, it is sufficient to prove a theorem from $\alpha_1 , \ldots , \alpha_n \rightarrow \beta$. Conversely, if $\alpha_1 \wedge \ldots \wedge \alpha_n \rightarrow \beta$ is proved then the theorem from $\alpha_1 , \ldots , \alpha_n$ infer β is assumed to be proved

Natural Deduction System (NDS)

Prove the theorem : infer $[(A \to B) \wedge (B \to C)] \to (A \to C)$

Solution: The Theorem is reduced to the theorem from $(A \to B),(B \to C)$ infer $(A \to C)$ using deduction theorem. Further to prove $A \to C$, we will have to prove a sub-theorem

from A infer C

Description	Formula	Comments	
Theorem	from $A \to B$, $B \to C$ infer $A \to C$	To be proved	
Hypothesis 1	$A \to B$	1	
Hypothesis 2	$B \to C$	2	
Sub-theorem	from A infer C	3	
Hypothesis	A		3.1
E : $\to$ (1,3.1)	B		3.2
E : $\to$ (2,3.2)	C		3.3
I : $\to$(3)	$A \to C$	Proved	

Proof of the Theorem

Axiomatic System

- It is based on a set of three axioms and one rule of deduction
- It is minimal in structure but powerful as the truth table and NDS approaches
- In axiomatic system, the proofs of the theorems are often difficult and require a guess in selection of appropriate axiom(s)
- In this system, only two logical operators **not(~)** and **implies(→)** are allowed to form a formula, other logical operators **V, Λ**, and ↔ can be easily expressed in terms of **~** and → using equivalence laws
- For example:

$A \land B \cong \sim(\sim A \lor \sim B) \cong \sim(A \rightarrow \sim B)$

$A \lor B \cong \sim A \rightarrow B$

$A \leftrightarrow B \cong (A \rightarrow B) \land (B \rightarrow A) \cong \sim[(A \rightarrow B) \rightarrow \sim(B \rightarrow A)]$

Axiomatic System

- In axiomatic system, there are three axioms, which are always true (or valid), and one rule called modus ponen (MP). Here α, β and γ are well-formed formulae of the axiomatic system. The three axioms and the rule are stated as follows:

  Axiom 1 : $\alpha \rightarrow (\beta \rightarrow \alpha)$

  Axiom 2 : $[\alpha \rightarrow (\beta \rightarrow \gamma)] \rightarrow [(\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma)]$

  Axiom 3 : $(\sim\alpha \rightarrow \sim\beta) \rightarrow (\beta \rightarrow \alpha)$

  Modus Ponen Rule Hypothesis : $\alpha \rightarrow \beta$ and α ; Consequent : β

  Interpretation of Modus Ponen Rule : Given that $\alpha \rightarrow \beta$ and α are hypotheses , β is inferred as a consequent

Axiomatic System

- Let $\Sigma = \{\alpha_1, \ldots, \alpha_n\}$ be a set of hypotheses. The formula α is defined to be a deductive consequence of Σ if either α is an axiom or a hypothesis or is derived from α_j, $1 \leq j \leq n$, using Modus Ponen inference rule. It is represented as $\{\alpha_1, \ldots, \alpha_n\} \vdash \alpha$ or more formally as $\Sigma \vdash \alpha$.
- If Σ is an empty set and α is deduced, then we can write $\vdash \alpha$. In this case, α is deduced from axioms only and no hypotheses are used. In such situations, α is said to be a theorem.

Axiomatic System

- Establish that $A \rightarrow C$ is a deductive consequence of $\{A \rightarrow B, B \rightarrow C\}$

  i.e. $\mathbf{\{A \rightarrow B, B \rightarrow C\} \vdash (A \rightarrow C)}$

Description	Formula	Comments
Theorem	$\{A \rightarrow B, B \rightarrow C\} \vdash (A \rightarrow C)$	Prove
Hypothesis 1	$A \rightarrow B$	1
Hypothesis 2	$B \rightarrow C$	2
Instance of Axiom 1	$(B \rightarrow C) \rightarrow [A \rightarrow (B \rightarrow C)]$	3
MP(2,3)	$[A \rightarrow (B \rightarrow C)]$	4
Instance of Axiom 2	$[A \rightarrow (B \rightarrow C)] \rightarrow [(A \rightarrow B) \rightarrow (A \rightarrow C)]$	5
MP(4,5)	$(A \rightarrow B) \rightarrow (A \rightarrow C)$	6
MP(1,6)	$(A \rightarrow C)$	Proved

Proof of the Theorem

Hence, we can conclude that $A \rightarrow C$ is a deductive consequence of $\{A \rightarrow B, B \rightarrow C\}$

Axiomatic System

Deduction Theorem : Given that Σ is a set of hypotheses and α and β are well formed formulae. If β is proved from $\{\Sigma \cup \alpha\}$, then according to the deduction theorem, $(\alpha \rightarrow \beta)$ is proved from Σ. Alternatively, we can write $\{\Sigma \cup \alpha\} \vdash \beta$ implies $\Sigma \vdash (\alpha \rightarrow \beta)$

Converse of Deduction Theorem : The converse of deduction theorem can be stated as:

Given $\Sigma \vdash (\alpha \rightarrow \beta)$, then $\{\Sigma \cup \alpha\} \vdash \beta$ is proved.

Points to remember while dealing with an axiomatic system:

- If α is given, then we can easily prove $\beta \rightarrow \alpha$ for any well-formed formulae α and β
- If $\alpha \rightarrow \beta$ is to be proved, then include α in the set of hypotheses Σ and derive β from the set $\{\Sigma \cup \alpha\}$. Then by using deduction theorem, we can conclude that $\alpha \rightarrow \beta$

Axiomatic System

Prove $\vdash \sim A \rightarrow (A \rightarrow B)$ by using deduction theorem

Solution: If we can prove $\{\sim A\} \vdash (A \rightarrow B)$ then using deduction theorem, we have proved

$\vdash \sim A \rightarrow (A \rightarrow B)$

Description	Formula	Comments
Theorem	$\{\sim A\} \vdash (A \rightarrow B)$	Prove
Hypothesis 1	$\sim A$	1
Instance of Axiom 1	$\sim A \rightarrow (\sim B \rightarrow \sim A)$	2
MP(1,2)	$(\sim B \rightarrow \sim A)$	3
Instance of Axiom 3	$(\sim B \rightarrow \sim A) \rightarrow (A \rightarrow B)$	4
MP(3,4)	$(A \rightarrow B)$	Proved

Proof of $\{\sim A\} \vdash (A \rightarrow B)$

Semantic Tableau System in Propositional Logic

- In both Natural Deduction and axiomatic systems, forward chaining approach is used for constructing proofs and derivations
- In axiomatic systems, we often require a guess for the selection of appropriate axiom(s) in order to prove a theorem
- Although the forward chaining approach is good for theoretical purposes, its implementation in derivations and proofs is difficult
- Two other approaches may be used : **Semantic Tableau** and **Resolution Refutation** methods, in both cases, proofs follow backward chaining approach
- In Semantic Tableau method, a set of rules are applied systematically on a formula or a set of formulae in order to establish consistency or inconsistency
- Semantic Tableau is a binary tree which is constructed by using semantic tableau rules with a formula as a root

Semantic Tableau Rules

Rule No.	Tableau tree
Rule 1	*$\alpha \wedge \beta$ is true if both α and β are true* $\alpha \wedge \beta$ | α | β
Rule 2	*$\sim(\alpha \wedge \beta)$ is true if either $\sim\alpha$ or $\sim\beta$ is true* $\sim(\alpha \wedge \beta)$ $\sim\alpha$ $\sim\beta$
Rule 3	*$\alpha \vee \beta$ is true if either α or β is true* $\alpha \vee \beta$ α β
Rule 4	*$\sim(\alpha \vee \beta)$ is true if both $\sim\alpha$ and $\sim\beta$ are true* $\sim(\alpha \vee \beta)$ | $\sim\alpha$ | $\sim\beta$
Rule 5	*$\sim(\sim\alpha)$ is true then α is true* $\sim(\sim\alpha)$ | α
Rule 6	*$\alpha \rightarrow \beta$ is true then $\sim\alpha \vee \beta$ is true* $\alpha \rightarrow \beta$ $\sim\alpha$ β
Rule 7	*$\sim(\alpha \rightarrow \beta)$ true then $\alpha \wedge \sim\beta$ is true* $\sim(\alpha \rightarrow \beta)$ | α | $\sim\beta$

α and β are two formulae

Semantic Tableau Rules
(Contd.)

Rule No.	Tableau tree
Rule 8	$\alpha \leftrightarrow \beta$ *is true then* $(\alpha \wedge \beta) \vee (\sim \alpha \wedge \sim \beta)$ *is true* $\alpha \leftrightarrow \beta$ $\alpha \wedge \beta$ $\quad$ $\sim\alpha \wedge \sim\beta$
Rule 9	$\sim(\alpha \leftrightarrow \beta)$ is *true* then $(\alpha \wedge \sim \beta) \vee (\sim \alpha \wedge \beta)$ *is true* $\sim(\alpha \leftrightarrow \beta)$ $\alpha \wedge \sim\beta$ $\quad$ $\sim\alpha \wedge \beta$

Semantic Tableau System in Propositional Logic

Construct a Semantic Tableau for a formula (A Λ∼B) Λ (∼B→ C)

Description	Formula	Line number
Tableau root	$(A \land \sim B) \land (\sim B \to C)$	1
Rule 1 (1)	$A \land \sim B$	2
	$\sim B \to C$	3
Rule 1 (2)	A	4
	$\sim B$	5
Rule 6 (3)	$\sim(\sim B)$ C	6
Rule 3 (6)	B √(open)	
	× (closed) $\{B, \sim B\}$	

Semantic Tableau System in Propositional Logic

- A path is said to be *contradictory* or *closed* (finished) whenever complementary atoms appear on the same path of a semantic tableau. This denotes inconsistency.
- If all paths of a tableau for a given formula α are found to be closed, it is called a *contradictory tableau*. This indicates that there is no interpretation or model that satisfies α.
- A formula α is said to be *satisfiable* if a tableau with root α is not a contradictory tableau, that is, it has at least one open path. We can obtain a model or an interpretation under which the formula α is evaluated to be true by assigning T (true) to all atomic formulae appearing on the open path of semantic tableau of α.
- A formula α is said to be *unsatisfiable* if a tableau with root α is a contradictory tableau.
- If we obtain a contradictory tableau with root $\sim\alpha$, we say that the formula α is *tableau provable*. Alternatively, a formula α is said to be *tableau provable* (denoted by $\vdash \alpha$) if a tableau with root $\sim\alpha$ is a contradictory tableau.
- A set of formulae $S = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ is said to *unsatisfiable* if a tableau with root $(\alpha_1 \land \alpha_2 \land ... \land \alpha_n)$ is a contradictory tableau.
- A set of formulae $S = \{\alpha_1, \alpha_2, ..., \alpha_n\}$ is said to be *satisfiable* if the formulae in a set are simultaneously true, that is, if a tableau for $\alpha_1 \land \alpha_2 \land ... \land \alpha_n$ has at least one open (or non-contradictory) path.
- Let S be a set of formulae. The formula α is said to be tableau provable from S (denoted by $S \vdash \alpha$) if there is a contradictory tableau from S with $\sim\alpha$ as a root.
- A formula α is said to be a *logical consequence* of a set S if and only if α is tableau provable from S.
- If α is tableau provable ($\vdash \alpha$) then it is also valid ($\models \alpha$) and vice versa.

Semantic Tableau System in Propositional Logic

Show that the formula α :(A Λ~B) Λ (~B→ C) is satisfiable

Solution: The Semantic Tableau has been drawn. We observe that there are two paths in it of which one path is closed, while the other is open.

This shows that (A Λ~B) Λ (~B→ C) is satisfiable

Semantic Tableau System in Propositional Logic

Show that α :(A ΛB) Λ (B→ ~A) is unsatisfiable using the tableau method

Solution: It can be proven that α :(A ΛB) Λ (B→ ~A) is unsatisfiable

Description	Formula	Line number
Tableau root	$(A \wedge B) \wedge (B \to \sim A)$	1
Rule 1 (1)	$A \wedge B$	2
	$B \to \sim A$	3
Rule 1 (2)	A	4
	B	5
Rule 6 (3)	$\sim B$ $\sim A$	
	$\times \{B, \sim B\}$ $\times \{A, \sim A\}$	

Semantic Tableau System in Propositional Logic

Consider a set S={~(AVB), (C→ B) , (A V C)} of formulae. Show that S is unsatisfiable

Solution: Consider the conjunction of formulae in the set as a root of Semantic Tableau.

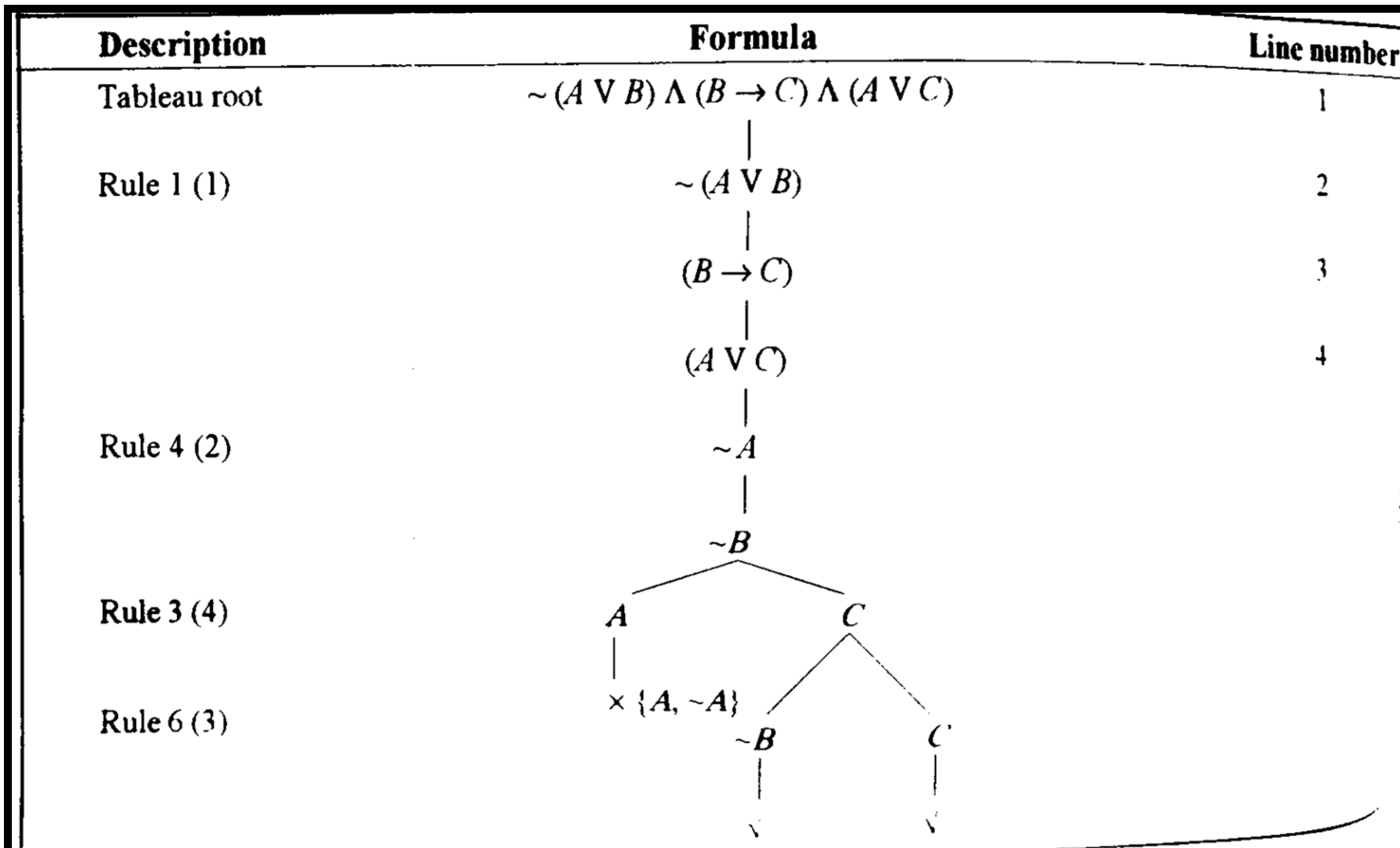
We can see that the tableau is contradictory, hence S is unsatisfiable

Description	Formula	Line number
Tableau root	$\sim(A \vee B) \wedge (C \rightarrow B) \wedge (A \vee C)$	1
Rule 1 (1)	$\sim(A \vee B)$	2
	$(C \rightarrow B)$	3
	$(A \vee C)$	4
Rule 4 (2)	$\sim A$	
	$\sim B$	
Rule 3 (4)	A ; C	
	$\times \{A, \sim A\}$	
Rule 6 (3)	$\sim C$; B	
	$\times \{C, \sim C\}$; $\times \{B, \sim B\}$	

Semantic Tableau System in Propositional Logic

Show that a set S={ ~(A V B), (B→ C) , (A V C) } is consistent.

Solution: Set S can be shown as consistent



We can construct a model for S by assigning truth value T to each literal on open path, i.e., {~A=T, ~B=T, C=T}

Semantic Tableau System in Propositional Logic

Show that B is a logical consequence of S= { A→ B, A }

Solution: Let us include ~B as a root with S in the tableau tree

Description	Formula	Line number
Tableau root	$\sim B$	1
Premise 1	$A \rightarrow B$	2
Premise 2	A	
Rule 6 (2)	$\sim A$ B	3
	$\times \{A, \sim A\}$ $\times \{B, \sim B\}$	

From table we see that B is tableau provable from S, that is, ~B as root gives contradictory tableau, thus B is a logical consequence of S

Semantic Tableau System in Propositional Logic

Show that α : B V ~(A → B) V ~ A is valid

Solution: In order to show that α is valid, we have to show that α is tableau provable, i.e., the tableau tree with ~α is contradictory . Table shows that α is a valid formula

Description	Formula	Line number
Tableau root	$\sim(B \vee \sim(A \to B) \vee \sim A)$	1
Rule 4 (1)	$\sim B$	2
	$\sim[\sim(A \to B) \vee \sim A]$	3
Rule 4 (3)	$\sim[\sim(A \to B)]$	4
	$\sim(\sim A)$	5
Rule 5 (5)	A	6
Rule 5 (4)	$A \to B$	
Rule 6 (7)	$\sim A$ B	7
	$\times \{A, \sim A\}$ $\times \{B, \sim B\}$	

Resolution Refutation method

- Another simple method that can be used in propositional logic to prove a formula or derive a goal from a given set of clauses by contradiction is the resolution refutation method.
- The term clause is used to denote special formula containing the boolean operators ~ and V
- Resolution Refutation is the most favoured method for developing computer-based systems that can be used to prove theorems automatically
- It uses a single inference rule, which is known as resolution based on modus ponen inference rule
- It is more efficient in comparison to NDS and Axiomatic System because in this case we do not need to guess which rule or axiom to apply in development of proofs
- Negation of the goal to be proved is added to the given set of clauses and using the resolution principle, it is shown that there is a refutation in the new set

Resolution Refutation method

Conversion of a Formula into a set of clauses

- In PC, there are two normal forms: DNF and CNF. A formula is said to be in its normal form if it is constructed using only natural connectives {~, Λ, V }
- Formally, a clause is defined as a formula of the form (L_1 V . . . V L_m). Therefore if a given formula is converted to its equivalent CNF as (C_1 Λ. . . Λ C_n) then the set of clauses is a set of each conjunct of CNF i.e. {C_1 . . . C_n }
- For eg. { A V B, ~A V D, C V ~ B} represent a set of clauses A V B, ~A V D, C V ~ B

Resolution Refutation method

Conversion of a Formula to its CNF

- Any formula in PC can be easily transformed into its equivalent CNF representation by using the following equivalence laws:
- Eliminate double negation by using $\sim(\sim A) \cong A$
- Use DeMorgan's Laws to push ($\sim$) negation immediately before the atomic formula

$$\sim(A \land B) \cong \sim A \lor \sim B$$

$$\sim(A \lor B) \cong \sim A \land \sim B$$

- Use distributive laws to get CNF

$$A \lor (B \land C) \cong (A \lor B) \land (A \lor C)$$

- Eliminate $\rightarrow$ and $\leftrightarrow$ by using the following equivalence laws:

$$A \rightarrow B \cong \sim A \lor B$$

$$A \leftrightarrow B \cong (A \rightarrow B) \land (B \rightarrow A)$$

Resolution Refutation method

Convert the formula $(\sim A \rightarrow B) \wedge (C \wedge \sim A)$ into its equivalent CNF representation

Solution: $(\sim A \rightarrow B) \wedge (C \wedge \sim A) \cong (\sim(\sim A) \vee B) \wedge (C \wedge \sim A)$

$\cong (A \vee B) \wedge (C \wedge \sim A)$

$\cong (A \vee B) \wedge C \wedge \sim A$

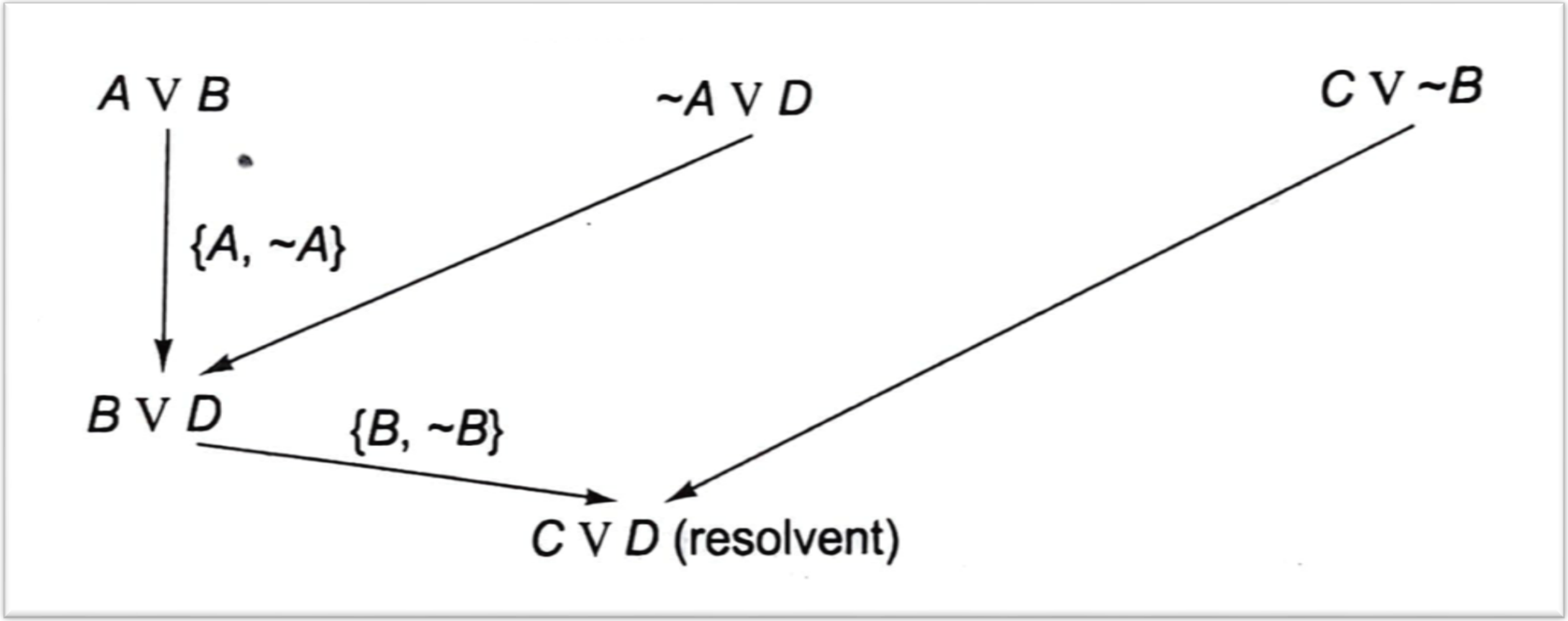
The set of clauses are $\{(A \vee B), C, \sim A\}$

Resolution of clauses

- Two clauses can be resolved by eliminating complementary pair of literals , if any, from both, a new clause is constructed by disjunction of the remaining literals in both the clauses
- If two clauses C_1 and C_2 contain a complementary pair of literals $\{ L, \sim L \}$ then these clauses may be resolved together by deleting L from C_1 and $\sim L$ from C_2 and constructing a new clause by the disjunction of remaining literals in C_1 and C_2. This new clause is called resolvent of C_1 and C_2. The clauses C_1 and C_2 are called parent clauses

Resolution Refutation method

Find resolvent of the clauses in the set {A V B, ~A V D, C V ~B }
Solution :



C V D is a resolvent of the set {A V B, ~A V D, C V ~B }

Resolution Refutation method

Key Points:

- If C is a resolvent of two clauses C_1 and C_2 then C is called a logical consequence of the set of the clauses $\{C_1, C_2\}$. This is known as Resolution Principle
- If a contradiction(or an empty clause) is derived from a set S of clauses using resolution then S is said to be unsatisfiable. Derivation of contradiction for a set S by resolution method is called a resolution refutation of S
- A clause C is said to be a logical consequence of S if C is derived from S
- Alternatively, using the resolution refutation concept, a clause C is defined to be a logical consequence of S if and only if the set S' = S ∪ {~C} is unsatisfiable, that is, a contradiction (or an empty clause) is deduced from the set S', assuming that initially the set S is satisfiable

Resolution Refutation method

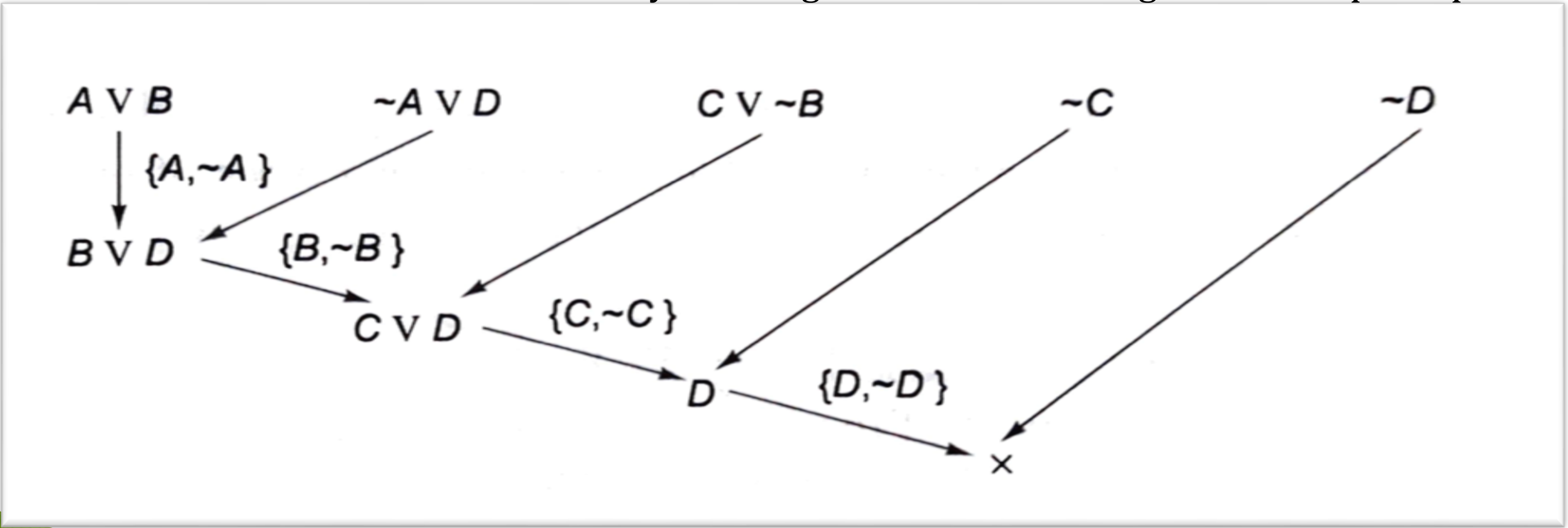
Using Resolution Refutation principle show that C V D is a logical consequence of

S = {A V B, ~A V D, C V ~B }

Solution : To prove the statement, first we will add negation of the logical consequence,

that is, ~(C V D) ≅ (~C Λ~D) to the set S to get S' = {A V B, ~A V D, C V ~B, ~C , ~D }.

We can show that S' is unsatisfiable by deriving contradiction using resolution principle



Since we get contradiction from S', we can conclude that (C V D) is a logical consequence of S = {A V B, ~A V D, C V ~B }

Predicate Logic

- Propositional Logic has many limitations. For eg. The facts *John is a boy*, *Paul is a boy*, and *Peter is a boy* can be symbolized by A,B and C respectively, in propositional logic but we cannot draw any conclusions about the similarities between A,B and C, i.e. we cannot conclude that these symbols represent boys
- Alternatively if we represent these facts as boy(John), boy(Paul), boy(Peter), then these statements give prima facie information that John, Paul, Peter are all boys. These facts can be easily generated from a general statement boy(X), where the variable X is bound with John, Paul, Peter . These facts are called instances of boy(X), while the statement boy(X) is called a predicate statement or expression.Here boy is a predicate symbol and X is its argument
- Statements like *All birds fly* cannot be represented by Propositional Logic. Such limitations are removed in Predicate Logic

Predicate Logic

- The Predicate Logic is a logical extension of propositional logic, which deals with the validity, satisfiability and unsatisfiability(inconsistency) of a formula along with the inference rules for derivation of a new formula

- Predicate Calculus is the study of predicate systems, when inference rules are added to predicate calculus, it becomes predicate logic

Predicate Calculus

Predicate Calculus has three more logical notations in addition to propositional calculus:

- **Term:** A term is defined as either a variable, or constant or n-place function. A function is defined as a mapping that maps n terms to a single term, a n-place function is $f(t_1, \ldots, t_n)$ where $t_1, \ldots, t_n$ are terms
- **Predicate :** A predicate as defined as a relation that maps n terms to a truth value {true, false}
- **Quantifiers:** Quantifiers are used with variables, universal, ∀ (for all) and existential quantifiers, ∃ (there exists)

Predicate Calculus

Well-formed formula : In Predicate Calculus, well-formed formula is defined as follows:

- Atomic formula $p(t_1, . . . ,t_n)$ is a well formed formula, where p is a predicate symbol and $t_1, . . ., t_n$ are the terms
- If α and β are well formed formulae, then ~(α), (α Λ β), (α V β), (α → β) and (α↔β) are also well formed formulae
- If α is a well formed formula and X is a free variable in α, then (∀X)α and (∃X)α are both well formed formulae. Here α is in scope of quantifier ∀ or ∃. Scope of the variable X is defined as that part of an expression where all occurrences of X have the same value
- Well formed formulae may be generated by applying the rules described above a finite number of times

First-Order Predicate Calculus

- If the quantification in predicate formula is only on simple variables and not on predicates or functions then it is called First-Order Predicate Calculus

  for eg. $\forall X \forall Y (p(X) \leftrightarrow p(Y))$

- If the quantification is over first-order predicates and functions, then it becomes Second-Order Predicate Calculus

  for eg. $\forall p (p(X) \leftrightarrow p(Y))$

- When inference rules are added to first-order predicate calculus, it becomes First-Order Predicate Logic (FOL)

Interpretation of Formulae in FOL

- $(\forall X)\ p(X)$ =true if and only if $p(X)$ is true , $\forall X \in D$ otherwise it is false
- $(\exists X)p(X)$ = true if and only if $\exists\ c \in D$ such that $p(c)$ is true otherwise it is false

Interpretation of Formulae in FOL

- Evaluate the truth value of a FOL formula $\alpha :(\forall X) (\exists Y) p(X,Y)$ under the following interpretation I : D={1,2}

  $p(1,1) = F, p(1,2) = T, p(2,1) = T, p(2,2) = F$

Solution: Let us denote true by T and false by F. For X=1, $\exists 2 \in D$ such that $p(1,2) = T$ and for X=2, $\exists 1 \in D$ such that $p(2,1) = T$. Hence , α is true under interpretation I

- Evaluate $\alpha :(\forall X) [p(X) \rightarrow q(f(X), c)]$ under the following interpretation :

  D= {1,2}
  c=1 (c is a constant from the domain D)
  f(1)=2 , f(2)=1
  p(1)=F, p(2)=T
  q(1,1)=T, q(1,2)=T, q(2,1)=F, q(2,2)=T

Solution: For X=1

$p(1) \rightarrow q(f(1), 1) \cong p(1) \rightarrow q(2, 1) \cong F \rightarrow F \cong T$

For X=2

$p(2) \rightarrow q(f(2), 1) \cong p(2) \rightarrow q(1, 1) \cong T \rightarrow T \cong T$

Hence α is true for all values of $X \in D$ under the interpretation I

Satisfiability and Unsatisfiability in FOL

- A formula α is said to be satisfiable if and only if there exists an interpretation I such that α is evaluated to be true under I. Alternatively, we may say that I is a model of α or I satisfies α
- A formula α is said to be unsatisfiable if and only if $\exists$ no interpretation that satisfies α or $\exists$ no model for α
- A formula α is said to be valid if and only if for every interpretation I, α is true
- A formula α is called a logical consequence of a set of formulae $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ if and only if for every interpretation I, if $\alpha_1 \Lambda\ \alpha_2 \Lambda \ldots \Lambda\ \alpha_n$ is evaluated to be true, then α is also evaluated to be true

* It is not possible to verify validity and unsatisfiability of a formula by evaluating it under infinite interpretations. This can solved in predicate logic by resolution refutation method.

Transformation of a formula into Prenex Normal Form(PNF)

- A formula is said to be in closed form if all the variables appearing in it are quantified and there are no free variables
- **Prenex Normal Form:** A closed formula α in FOL is said to be in PNF if and only if α is represented as $(Q_1X_1)\ (Q_2X_2)\ \dots\ (Q_nX_n)\ M$, where Q_k are quantifiers (∀ or ∃), X_k are variables for $1 \leq k \leq n$, while M is a formula free from quantifiers. The list of quantifiers $[(Q_1X_1)\ \dots\ (Q_nX_n)]$ is called prefix and M is called the matrix of a formula α. Here M is assumed to be represented in CNF notation.
- For eg. (∃X)(∀ Y) [p(X) V q(X,Y)] is in PNF notation, whereas

  (∀ X) [p(X) →(∃Y)q(X,Y)] is not in PNF notation

Transformation of a formula into Prenex Normal Form(PNF)

Conversion of formula into PNF notation

- A formula can be easily transformed or converted into PNF using various equivalence laws.
- Some conventions used: α – FOL formula α without a variable X
  α[X] – FOL formula α with a variable X
  Q – Quantifier(∀ or ∃)

Equivalence Laws: (in addition to Propositional Logic) [* is V or Λ]

Law 1: (QX) α[X] * β ≅ (QX) (α[X] * β)

Law 2: α * (QX) β[X] ≅ (QX)(α * β[X])

Law 3: ~(∀X) α[X] ≅ (∃X) (~ α[X])

Law 4: ~(∃X) α[X] ≅ (∀X) (~ α[X])

Law 5: (∀X) α[X] Λ (∀X) β[X] ≅ (∀X) (α[X] Λ β[X])

Law 6: (∃ X) α[X] V (∃ X) β[X] ≅ (∃ X) (α[X] V β[X])

Conversion of PNF to its standard form (Skolemization)

- The prenex normal form of a given formula can be further transformed into a special form called skolemization or standard form. This form provides clauses which can then be used in resolution process
- The process of eliminating existential quantifiers from the prefix of a PNF notation and replacing the corresponding variable by a constant or a function is called skolemization, such a constant or function is called skolem constant or skolem function respectively

Conversion of PNF to its standard form (Skolemization)

Skolemization Procedure

- Let (Q_1X_1) (Q_2X_2) … (Q_nX_n) M be the PNF notation corresponding to some formula, where (Q_1X_1) … (Q_nX_n) represent prefix, while M denotes a matrix

Skolemization Steps:

- Scan prefix from left to right till we obtain the first existential quantifier
  - If Q_1 is the first existential quantifier then choose a new constant c ∉ { set of constants in M }. Replace all occurrences of X_1 appearing in matrix M by c and delete $(Q_1 X_1)$ from the prefix to obtain new prefix and matrix
  - If Q_r is the first existential quantifier and $Q_1 . . . Q_{r-1}$are universal quantifiers appearing before Q_r , then choose a new (r-1) place function symbol f ∉ {set of functions appearing in M}. Replace all occurrences of X_r in M by $f(X_1, . . . , X_{r-1})$ and remove (Q_rX_r) from prefix
- Repeat the process till all existential quantifiers are removed from M

Clauses in FOL

- A clause is defined as a closed formula written in the form (L_1 V . . . V L_m), where each L_i is a literal and all variables occurring in $L_{1,}$ ... , L_m are universally quantified
- Let S={$C_{1,}$ ... , C_m } be a set of clauses that represents a standard form of a given formula α. Then the following definitions hold true:
  - A formula α is said to be unsatisfiable if and only if its corresponding set S is unsatisfiable
  - S is said to be unsatisfiable if and only if there ∃ no interpretation that satisfies all the clauses of S simultaneously
  - S is said to be satisfiable if and only if each clause is satisfiable, i.e. , ∃ an interpretation that satisfies all the clauses of S simultaneously
  - Alternatively , an interpretation I is said to model S if and only if I models each clause of S

Resolution Refutation Method in FOL

- Resolution Refutation Method in FOL is used to test unsatisfiability of a set (S) of clauses corresponding to the predicate formula. It is an extension of the Resolution Refutation Method in Propositional Logic
- Find the resolvent of two clauses CL_1 and CL_2, where p, q, r are predicate symbols, X is a variable and f is a unary function

$$CL_1 = p(X) \vee q(X)$$

$$CL_2 = \sim p(f(X)) \vee r(X)$$

Solution: If we substitute f(a) for X in CL_1 and a for X in CL_2, where a is a new constant from the domain, then we obtain

$$CL_3 = p(f(a)) \vee q(f(a))$$

$$CL_4 = \sim p(f(a)) \vee r(a)$$

We observe that CL_3 and CL_4 have complementary literals as $p(f(a))$ and $\sim p(f(a))$

Resolvent of CL_3 and CL_4 is $q(f(a)) \vee r(a)$

Resolution Refutation Method in FOL

Show that the formula $\alpha : \forall (X)(p(X) \wedge \sim[q(X) \rightarrow p(X)])$ is unsatisfiable

Solution:

$$p(X) \wedge \sim[q(X) \rightarrow p(X)] \cong p(X) \wedge \sim[\sim q(X) \vee p(X)]$$

$$\cong p(X) \wedge \sim\sim q(X) \wedge \sim p(X)$$

$$\cong p(X) \wedge q(X) \wedge \sim p(X)$$

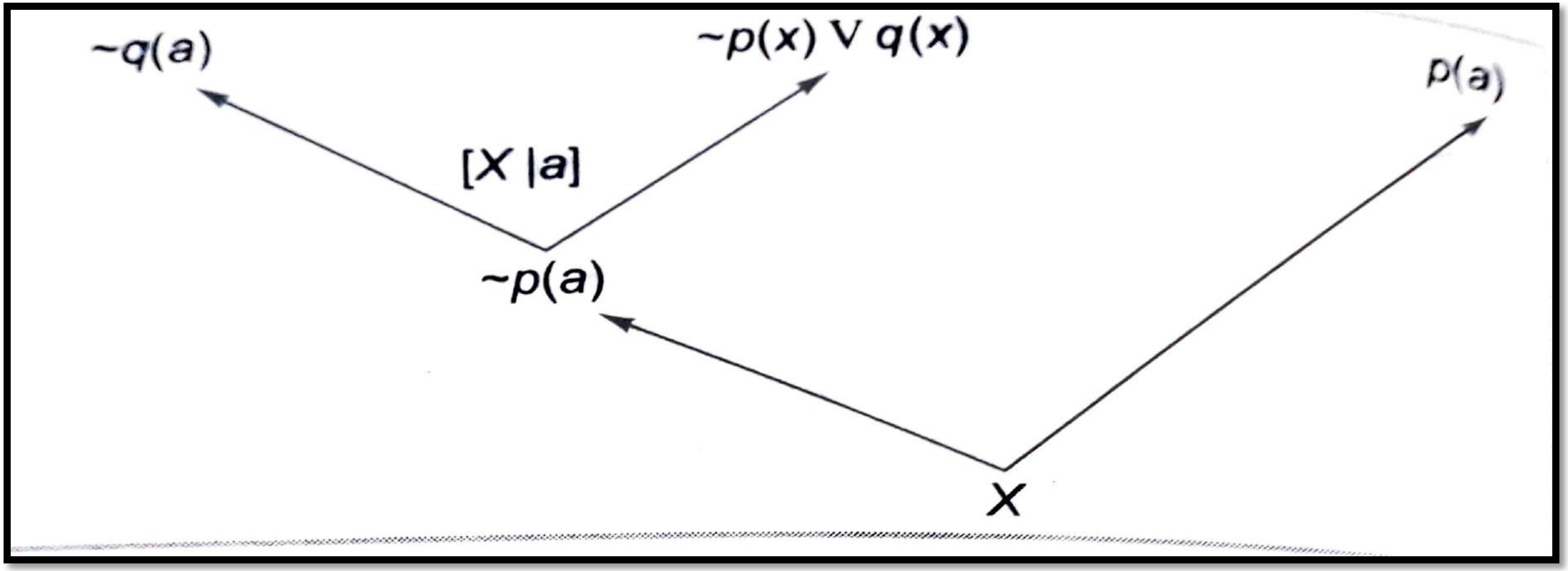
The set of clauses is written as S = $\{p(X), q(X), \sim p(X)\}$. Since there is a contradiction in S itself , S is unsatisfiable and consequently α is unsatisfiable

Resolution Refutation Method in FOL

Show that q(a) is a logical consequence of formulae α and β where

α : ∀ (X)[p(X) →q(X)]

β : p(a)



We conclude that q(a) is a logical consequence of formulae α and β