2.1.3 Theorem Proving by Propositional Logic:

We present here two techniques for logical theorem proving in propositional logic. These are : *(1) Semantic methods* , *(2) Syntactic methods* of theorem proving.

(1) Semantic Method for Theorem Proving:

The following notation will be used to represent a symbolic theorem, stating that conclusion "**C**" follows from a set of premises $p_1, p_2, \ldots, p_n$

$$p_1, p_2, \ldots, p_n \Rightarrow C$$

In this technique, we first construct a truth table representing the relationship of p_1 through p_n with "**c**". Then we test the validity of the theorem by checking whether both the ***forward and backward chaining methods***, to be presented shortly, hold good. The concept can be best illustrated with an example.

Example 1: Let us redefine p_1= ***the-sky-is-cloudy***, p_2 = ***it-will-rain*** and $p_3 \equiv p_1 \rightarrow p_2$ to be three propositions. We now form a truth table of p_1, p_2 and p_3 , and then attempt to check whether forward and backward chaining holds good for the following theorem: $p_1, p_3 \Rightarrow p_2$, We have $p_3 \equiv p_1 \rightarrow p2 \equiv \neg p_1 \vee p_2$

p_1	p_2	p_3 $(\neg p_1 \vee p_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Fig.(2.3): Truth Table of p_1, p_2, p_3

Forward chaining: *When all the* ***premises*** *are* ***true****, check whether the* ***conclusion*** *is* ***true****. Under this circumstance, we say that forward chaining holds good.*

In this example, when p_1 and p_3 are true, check if p_2 is true. Note that in the last row of the truth table, $p_1 = 1$, $p_3 = 1$ yield $p_2 = 1$. So, forward chaining holds good. Now, we test for backward chaining.

Backward chaining**:** *When all the* ***consequences*** *are* ***false****, check whether at least one of the* ***premises*** *is* ***false****.*

In this example $p_2=0$ in the first and third row. Note that when $p_2=0$, then $p_1=0$, in the first row and $p_3=0$ in the third row. So, backward chaining holds good. As forward and backward chaining both are satisfied together, the theorem: $p_1, p_3 \Rightarrow p_2$ also holds good.

Example 2: Show that for **example** 1, $p_2, p_3 \Rightarrow p_1$. It is clear from the truth **table 2** that when $p_1=0$, then $p_2=0$ (first row) and $p_3 = 1$ (first row), backward chaining holds good. But when $p_2= p_3 =1$, $p_1=0$ (second row), forward chaining fails. Therefore, the theorem does not hold good.

(2). Syntactic Methods for Theorem Proving:

Before presenting the syntactic methods for theorem proving in propositional logic, we state a few well-known theorems.

Standard theorems in propositional logic:

Assuming **p, q** and **r** to be propositions, the list of the standard theorems is:

1. $p, q \Rightarrow p \wedge q$
2. $p, p \rightarrow q \Rightarrow q$ **(Modus Ponens)**
3. $\sim p, p \vee q \Rightarrow q$
4. $\sim q, p \rightarrow q \Rightarrow \sim p$ **(Modus Tollens)**
5. $p \vee q, p \rightarrow r, q \rightarrow r \Rightarrow r$
6. $p \rightarrow q, q \rightarrow r \Rightarrow p \rightarrow r$ **(chaining)**
7. $p, p \rightarrow q, q \rightarrow r \Rightarrow r$ **(Modus Ponens & chaining)**
8. $p \vee (q \wedge \sim q) \Leftrightarrow p$
9. $p \wedge (q \vee \sim q) \Leftrightarrow p$
10. $p \rightarrow q \Leftrightarrow \sim p \vee q$
11. $\sim(p \rightarrow q) \Leftrightarrow p \wedge \sim q$
12. $p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
13. $p \leftrightarrow q \Leftrightarrow (p \wedge q) \vee (\sim p \wedge \sim q)$
14. $p \rightarrow (q \rightarrow r) \Leftrightarrow (p \wedge q) \rightarrow r$
15. $p \rightarrow q \Leftrightarrow \sim q \rightarrow \sim p$ **(contraposition theorem)**

The **syntactic approach** for theorem proving can be done in two ways:

(a) Substitution Method (b) by Wang's Method.

(a) Method of Substitution

By this method, **left-hand side** (or **right-hand side**) of the statement to be proved is chosen and the standard formulas, presented above, are applied selectively to prove the other side of the statement.

Example 3: Prove the contraposition theorem. The contraposition theorem can be stated as follows. When **p** and **q** are two propositions, the theorem takes the form of

$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$

Now , L.H.S = $p \rightarrow q$

$\Rightarrow \sim p \vee q$ [by (10)]

$\Rightarrow q \vee \sim p$

$\Rightarrow \sim(\sim q) \vee \sim p$

$\Rightarrow \sim q \rightarrow \sim p$ = R.H.S.

Analogously, starting with the **R.H.S,** we can easily reach the **L.H.S**. Hence, the theorem bi-directionally holds good.

Example 4: Prove theorem by method of substitution.

$p \rightarrow (q \rightarrow r) \Leftrightarrow (p \wedge q) \rightarrow r$

Proof: **L.H.S = $p \rightarrow (q \rightarrow r)$**

$\Rightarrow p \rightarrow (\sim q \vee r)$ [by (10)]

$\Rightarrow \sim p \vee (\sim q \vee r)$ [by (10)]

$\Rightarrow (\sim p \vee \sim q) \vee r$

$\Rightarrow \sim(p \wedge q) \vee r$ by De Morgan's law

$\Rightarrow (p \wedge q) \rightarrow r$ = R.H.S [by (10)]

Analogously, the **L.H.S**. can be equally proved from the **R.H.S.** Hence, the theorem follows bi-directionally.

(b) Theorem Proving by Using Wang's Algorithm:

Any theorem of propositional logic is often represented in the following form:

$$p_1, p_2, \ldots p_n \Rightarrow q_1, q_2, \ldots, q_m$$

where p_i and q_j represent propositions. The comma in the **L.H.S**. represents AND operator, while that in the **R.H.S**. represents OR operator.

Writing symbolically, $p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q_1 \vee q_2 \vee \ldots \vee q_m$

This kind of theorem can be easily proved using Wang's algorithm. The algorithm is formally presented below.

Wang's algorithm

Step I: Starting condition: Represent all sentences, involving only $\wedge$, $\vee$ and $\neg$ operators.

Step II: Recursive procedure: Repeat steps (a), (b) or (c) whichever is appropriate until the stopping condition, presented in **step III**, occurs.

a) **Negation Removal:** In case negated term is present at any side (separated by comma) bring it to the other side of implication symbol without its negation symbol.

e.g., p, q, $\neg$ r $\Rightarrow$ s $\vdash$ p, q $\Rightarrow$ r, s

b) **AND, OR Removal**: If the L.H.S. contains $\wedge$ operator, replace it by a comma. On the other hand if R.H.S. contains $\vee$ operator, also replace it by a comma. **e.g., p $\wedge$ r, s $\Rightarrow$ s $\vee$ t $\vdash$ p, r, s $\Rightarrow$ s, t**

c) **Theorem splitting:** If the L.H.S. contains OR operator, then split the theorem into two sub-theorems by replacing the OR operator. Alternatively, if the R.H.S. contains AND operator, then also split the theorem into two sub-theorems.

e.g., p $\vee$ r $\Rightarrow$ s, t $\vdash$ p $\Rightarrow$ s, t & r $\Rightarrow$ s, t : Sub-theorems

e.g., p, r $\Rightarrow$ s $\wedge$ t $\vdash$ p, r $\Rightarrow$ s & p, r $\Rightarrow$ t : Sub-theorems

Step III: Stopping Condition: Stop theorem proving process if either of (a) or (b) occurs.

(a) If both L.H.S. and R.H.S. contain common atomic terms, then stop.

(b) If L.H.S. and R.H.S. have been represented as a collection of atomic terms, separated by commas only and there exist no common terms on both sides, then stop.

End of Algorithm.

In case all the sub-theorems are stopped, satisfying condition III (a), then the theorem holds good. We would construct a tree structure to prove theorems using Wang's algorithm. The tree structure is necessary to break each theorem into sub-theorems.

Example 5: Prove the chaining rule with Modus Ponens using Wang's algorithm.

Proof: The chaining rule with Modus Ponens can be described as:

p, p → q, q → r $\Rightarrow$ r where p, q and r are propositions (atomic).

We now construct the tree. A node in the tree denotes one propositional expression. An arc in the tree denotes the step of Wang's algorithm, which is applied to produce the next step. The bold symbols in both the left- and right-hand side of the implication symbol

describe the termination of the sub-tree:

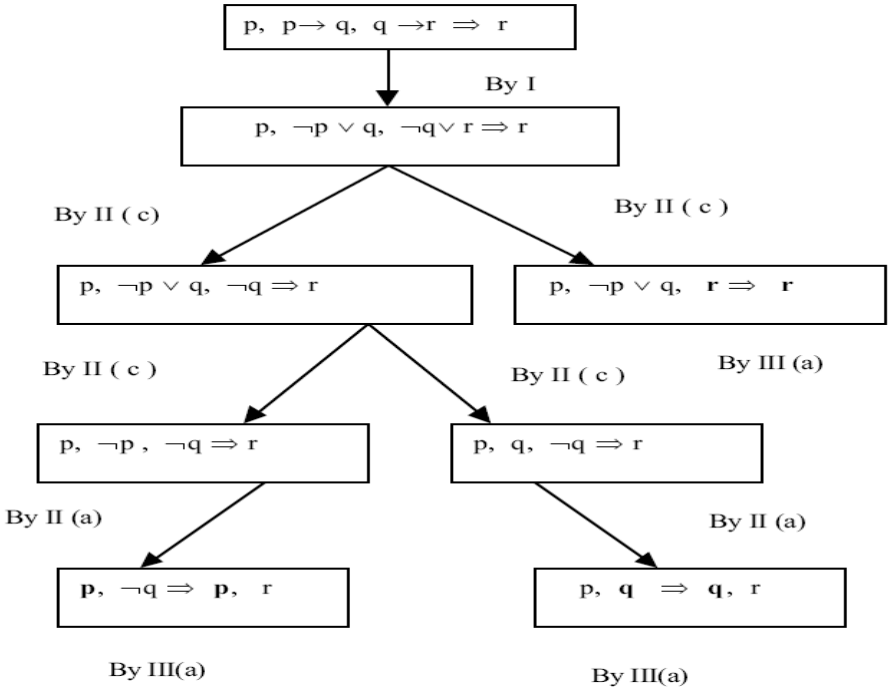

Fig. (2.4): Tree used to prove a propositional theorem by Wang's algorithm

Note*: Since all the terminals of the tree have been stopped by using III (a), the theorem* ***holds good****.*