**32002: AI (Propositional Logic, Syntax, semantics, & Inferencing)** **Spring 2015**

Lecture 2: January 05, 2015

*Lecturer: K.R. Chowdhary* *: Professor of CS (VF)*



## 2.1 Logic

Now, though mathematical logic remains the branch of pure mathematics, it is extensively applied to computer science and artificial intelligence in the form of propositional logic and predicate logic (first order predicate logic (FOPL)).

As per the Newell's and Simons's Physical Symbol System Hypothesis (PSSH), discussed in the previous chapter, knowledge representation is first requirement of achieving intelligence. This chapter presents the knowledge representation using *propositional logic*, introduces first order predicate logic (FOPL), and drawing of inferences using propositional logics.

Logic is formal method for reasoning, using it concepts can be translated into symbolic representation, which closely approximate the meaning of these concepts. The symbolic structures can be manipulated using computer programs to deduce facts to carry out the form of automated reasoning.

The *aim* of logic is to learn principles of valid reasoning as well as to discern good reasoning from bad reasoning, identifying invalid arguments, distinguishing *inductive* v/s *deductive* arguments, identifying *fallacies* as well avoiding the fallacies.

We study the logic because of the following reasons:

- Logic deals with what follows from what? For example, Logical consequence, inference pattern, and validating such patterns,
- We want the computer to understand our language and does some intelligent tasks for us (Knowledge representation),
- To engage in debates, solving puzzles, game like situation,
- Identify which one is a fallacious argument and what is type of fallacy?
- Proving theorems through *deduction*. To find out whether what ever proved is correct, or whatever obviously true has a proof?, and
- Some problems concerning the foundations of mathematics.

## 2.2 Propositional Logic

The propositional logic deals with individual Propositions, which are viewed as *atoms*, i.e., these cannot be further broken down into smaller constituents. For building propositional logic, first we describe the

logic with the help of a formula called *Well-Formed Formulas* (wff read as woofs). A formula is a *syntactic* concept, which means whether or not a string of symbols is a formula not. It can be determined solely based on its formal construction, i.e., whether it can be built according to its construction rules. Therefore, we are in a position to verify that a sequence of symbols is a formula or not, as per the specified rules. This function of verification, in a compiler is done by a *parser* - to verify whether the formula belong to particular programming language or not. A parser also constructs a parse tree of the given formula through which it tells how the formula is constructed.

The meaning (*semantics*) is associated with each formula by defining its *interpretation*, which assign a value *true* ($T$) or *false* ($F$) to every formula. The syntax is also also used to define the concept of *proof* - the symbolic manipulations of formulas to deduce the given theorem. The important thing we should note is that provable formulas are only those which are always true.

Let following be the propositions:

$p$ = Sun is star.

$q$ = Moon is satellite.

We can construct the following formulas using the above propositions:

$p \wedge q$ = Sun is star *and* Moon is satellite.

$p \vee q$ = Sun is star *or* Moon is satellite tennis.

$\neg p \vee q$ = Sun is *not* star or Moon is satellite.

$\neg p \rightarrow q$ = *if* Moon is *not* star then Moon is satellite.

A formula in propositional logic can be recursively defined as follows:

i) Each propositional variable and null are formulas, therefore, $p, q, \phi$ are formulas,

ii) If $p, q$ are formulas, then $p \wedge q, p \vee q, \neg p, p \rightarrow q, (p)$, are also formulas,

iii) A string of symbols is a formula only as determined by finitely many applications of above (i) and (ii), and

iv) nothing else is propositional formula.

This recursive form of the definition can be expressed using $BNF$ (Backups-Naur Form) notation as follows:

$$\begin{aligned} &1.\ formula := atomicformula \mid \neg formula \mid formula \wedge formula \\ &\quad \mid formula \vee formula \mid formula \rightarrow formula \mid (formula) \\ &2.\ atomicformula := \bot \mid p \mid q \mid r \mid p_0 \mid p_1 \mid p_2 \mid \ldots \end{aligned} \tag{2.1}$$

In the above notation, the symbols $formula$, $atomicformula$, that appear to the left-hand, called *non-terminals* and represent grammatical classes, and $p, q, r, \bot, p_1$, etc, that appear only to the right-hand side, are called *terminals*, and represent the symbols of the language.

A sentence in the propositional language is obtained through a derivation that starts with a non-terminal, and repeatedly applied the substitution rules from the BNF notations, until the terminals are reached.

**Example 2.1** *Derivation for* $p \wedge q \rightarrow r$.

The sequence of substitutions rules to derive this formula, i.e., to establish that it is syntactically correct, are as follows:

$$\begin{aligned} formula &\Rightarrow formula \wedge formula \\ &\Rightarrow formula \wedge formula \rightarrow formula \\ &\Rightarrow atomic \wedge formula \rightarrow formula \\ &\Rightarrow p \wedge formula \rightarrow formula \\ &\Rightarrow p \wedge atomic \rightarrow formula \\ &\Rightarrow p \wedge q \rightarrow formula \\ &\Rightarrow p \wedge q \rightarrow atomic \\ &\Rightarrow p \wedge q \rightarrow r \end{aligned}$$

The symbol *atomic* stands for atomic-formula, and the symbol $\Rightarrow$ stands for "implies", i.e., the expression to right to this is implied by the previous expression.

The derivation can also be represented by a *derivation-tree* (*parse-tree*), shown in figure 2.1. From the derivation-tree, we can obtain another tree shown in figure 2.2, called *syntax-tree* or *formation-tree*, by replacing each non-terminal by the child that is an operator under that. There is always unique syntax-tree for any given formula.

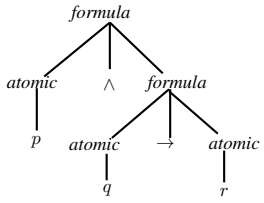


Figure 2.1: Parse-tree for the expression $p \wedge q \rightarrow r$.

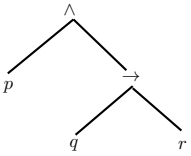


Figure 2.2: Syntax-tree for the expression $p \wedge q \rightarrow r$.

□

Considering two propositions $p, q$, the interpretation (semantics) of the formulas constructed when they are joined using binary operators $\vee, \wedge$, and $\rightarrow$ are shown in table 2.1.

Table 2.1: Interpretation of propositional formulas.

| $p$ | $q$ | $p \vee q$ | $p \wedge q$ | $p \rightarrow q$ |
|---|---|---|---|---|
| $F$ | $F$ | $F$ | $F$ | $T$ |
| $F$ | $T$ | $T$ | $F$ | $T$ |
| $T$ | $F$ | $T$ | $F$ | $F$ |
| $T$ | $T$ | $T$ | $T$ | $T$ |

The *Material conditional* '$\rightarrow$' joins two simpler propositions, , e.g., $p \rightarrow q$, read as "if $p$ then $q$". The proposition to the left of the arrow is called the *antecedent* and to the right is *consequent* (There is no such designation for conjunction or disjunction operators because they are commutative operations). It expresses that $q$ is true whenever $p$ is true. Thus it is true in every case in table 2.1, except in row three, because this is the only case when $p$ is true but $q$ is not. Using "if $p$ then $q$" we can expresses that "if it is raining outside then there is a cold over Kashmir". The material conditional is often confused with *physical causation*. The material conditional, however, only relates two propositions by their truth-values - which is not the relation of *cause* and *effect*. It is contentious in the literature whether the material implication represents logical causation.

## 2.3 Interpretation of Formulas

The *logical consequence* is the central concept in the foundations of logic. It is much more interesting to assume that a set of formulas is true and then to investigates the consequences of these assumptions.

Assume that $\theta$ and $\psi$ are formulas (sentences) of a set $\mathcal{P}$, and $\mathcal{I}$ is an interpretation of $\mathcal{P}$. The sentence $\theta$ of propositional logic is true under an interpretation $\mathcal{I}$ iff $\mathcal{I}$ assigns the truth value $T$ to that sentence. If a sentence is true under an interpretation, then that interpretation is called a *model* of that sentence. The $\theta$ is false under an interpretation $\mathcal{I}$ iff $\theta$ is not true under $\mathcal{I}$.

A sentence is logically true (valid) iff it is true under every interpretation. $\models \theta$ means that $\theta$ is valid.

A sentence $\psi$ of propositional logic is a *logical consequence* of a sentence $\theta$, represented as $\theta \models \psi$, if every interpretation $\mathcal{I}$ that satisfy $\theta$ also satisfy $\psi$. In fact, $\psi$ need not to be true in every possible interpretation, only in those interpretations which satisfy $\theta$, i.e, those interpretations which satisfy every formula in $\theta$. In the formula $((p \rightarrow q) \wedge p) \vdash q$, the $q$ is logical consequence of $((p \rightarrow q) \wedge p)$. The sign '$\vdash$' is sign of deduction.

A sentence of propositional logic is *consistent* iff it is true under at least one interpretation. It is *inconsistent* if it is not consistent.

**Definition 2.2** *Logical consequence.*

Given $S$ as set of formulas and $A$ as a formula, if $A$ is *true* in all the models of $S$, then $A$ is logical consequence of $S$, we denote it by $S \models A$. □

**Example 2.3** *Determine the logical consequence of* $\psi = (p \vee r) \wedge (\neg q \vee \neg r)$ *from* $\theta = \{p, \neg q\}$ *and validity for* $\psi$.

**Solution.** Here $\psi$ is logical consequence of $\theta$, denoted by $\theta \models \psi$, because $\psi$ is true under all the interpretations such that $v(p) = True$, and $v(q) = False$, the interpretation for which $\theta$ is satisfied. Here $v$ stands for

'interpretation of'.

However, $\psi$ is not valid, since it is not true under the interpretation $v(p) = F, v(q) = T, v(r) = T$.

□

## 2.4 Syntax and Semantics

*Syntax* is name given to a correct *structure* of a statement. It is meaning associated with the expression. Its mapping to the real-world situation is *semantics*. The semantics of a language defines the truth of each sentence with respect to each possible *world*. For example, the usual semantics for interpretation of the statement $(p \vee q) \wedge r$ is true in a *world* where either $p$ or $q$ or both are *true* and $r$ is *true*. Different worlds can be all the possible sets of *truth* values of $p, q, r$. The truth values are simply the assignment to these variables, and not necessarily the values which are only *true*. For example, $v(p) = F, v(q) = F, v(r) = T$; and $v(p) = T, v(q) = F, v(r) = T$ are the worlds for the expression $(p \vee q) \wedge r$.

## 2.5 Semantic Tableau

Semantic tableau is relatively efficient method for deciding satisfiability for formula of propositional calculus. The method (or algorithm) *systematically* searches for a model for a formula. If it is found the formula is satisfiable, else not satisfiable. We start with the definition of some terms, and then analyze some formulas to motivate us for construction of semantic tableau.

**Definition 2.4** *Literal and complementary pair of literals.*

A literal is an atom or negation of an atom. For any atom $p$, the set $\{p, \neg p\}$ is called complementary pair of literals. For any formula $A$, $\{A, \neg A\}$ is complementary pair of formulas. □

**Example 2.5** *Analysis of satisfiability of a formula.*

Consider that a formula $A = p \wedge (\neg q \vee \neg p)$, has an arbitrary interpretation $v$. Given this, $v(A) = T$ iff $v(p) = T$ and $v(\neg q \vee \neg p) = T$. Hence, $v(A) = T$ iff either

1. $v(p) = T$ and $v(\neg q) = T$, or
2. $v(p) = T$ and $v(\neg p) = T$.

Hence $A$ is satisfiable, if either (1) interpretation holds or (2) holds. But (2) is not feasible. So, $A$ is satisfiable when interpretation of (1) holds true. Note that the satisfiability of a formula is reduced to the satisfiability of literals.

It is clear that a set of literals is satisfied if and only if it does not contain complementary pair of literals. In the above case, the pair of literals $\{p, \neg p\}$ in case (2) is complementary pair, hence the formula is unsatisfied for this interpretation. However, the first set $\{p, \neg q\}$ is not complementary pair, hence it is satisfiable.

From the above discussion, we have trivially constructed a model for the formula $A$ by assigning *True* to positive literals and *False* to negative literals. Hence, $p$ = *True*, and $q$ = *False* makes the set in (1) true, hence $\{p = T, q = F\}$ is a model for formula $A$.

The above is a *search process*, and can be represented by a tree shown in figure 2.3. The leaves in the tree represent a set of literals that must be satisfied. A leaf containing complementary pair of literals is marked *closed* by $\times$, while the satisfying leaf is marked as *open* by $\odot$.

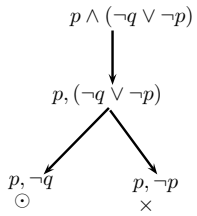


Figure 2.3: Tree for Semantic tableau.

The construction process of tree can be represented as an algorithm, to find out if some model exists for a formula, and what is that model. □

**Definition 2.6** *Semantic Tableau.*

*Semantic Tableau* is a tree, each node of which will be labeled with a set of formulas, and these formulas are inductively expanded to leaves such that each leaf is marked as *open* by $\odot$ or *closed* by $\times$. □

**Definition 2.7** *Completed tableau.*

A semantic tableau whose construction is terminated is called *completed tableau*. A completed tableau is *closed* if all leaves are marked *closed*. Otherwise it is *open* i.e., some leaves are open. □

**Example 2.8** *Find out whether* $(p \vee q) \wedge (\neg p \wedge \neg q)$ *is satisfiable, using tableau method.*

**Solution.** Let $A = (p \vee q) \wedge (\neg p \wedge \neg q)$. For the satisfaction of $A$, $v(A) = True$ for some assignments. That is, $v(p \vee q) = True$ and $v(\neg p \wedge \neg q) = True$. Thus, $v(A)$ is *True* if either,

- $v(p) = T, v(\neg p) = True, v(\neg q) = True$, or
- $v(q) = True, v(\neg p) = True, v(\neg q) = True$.

So that, two sets of literals are,

$(p, \neg p, \neg q)$ and $(q, \neg p, \neg q)$.

Since both contain complementary pairs, hence neither of the literals is satisfiable. So it is impossible to find model for $A$, and $A$ is unsatisfiable. □

## 2.6 Reasoning Patterns

The reasoning pattern comprise *inference methods*: *modus ponens, modus tollens, syllogism*; and *Proof methods: resolution theorem, model checking, Normal forms*, and *conversions between normal forms.*

Table 2.2: Inference Rules.

| **Rule** | **Formula** | **Description** |
|---|---|---|
| *Modus Ponens* | $((p \to q) \wedge p) \vdash q$ | If $p$ then $q$; $p$; therefore $q$ |
| *Modus Tollens* | $((p \to q) \wedge \neg q) \vdash \neg p$ | If $p$ then $q$; not $q$; therefore not $p$ |
| *Hypothetical Syllogism* | $((p \to q) \wedge (q \to r)) \vdash (p \to r)$ | if $p$ then $q$; if $q$ then $r$; therefore, if $p$ then $r$ |
| *Disjunctive Syllogism* | $((p \vee q) \wedge \neg p) \vdash q$ | Either $p$ or $q$, or both; not $p$; therefore, $q$ |

Some of the deductions through inferences, in the propositional logic are shown in table 2.2. The enumeration table of all possible worlds of a *modus ponens* is shown in table 2.3. We note that it is valid inference, as the sentence $((p \to q) \wedge p) \to q$, when $q$, the inference implied, is true in all the rows.

Table 2.3: Modus ponens is valid inference.

| $p$ | $q$ | $p \to q$ | $(p \to q) \wedge p$ | $((p \to q) \wedge p) \to q$ |
|---|---|---|---|---|
| $F$ | $F$ | $T$ | $F$ | $T$ |
| $F$ | $T$ | $T$ | $F$ | $T$ |
| $T$ | $F$ | $F$ | $F$ | $T$ |
| $T$ | $T$ | $T$ | $T$ | $T$ |

### 2.6.1 Resolution

The *resolution rule* is an inference which uses *deduction* approach. It is used in theorem proving. If two disjunctions have complementary literals, then a resultant inference of these is disjunction of these expressions, with complementary terms removed. If $p = p_1 \vee p_2 \vee c$ and $q = q_1 \vee \neg c$ are two formulas, then resolution of $p$ and $q$ results to dropping of $c$ and $\neg c$ and disjunction is performed of the remaining propositions of $p$ and $q$, as follows:

$$\frac{(p_1 \vee p_2 \vee c), (q_1 \vee \neg c)}{p_1 \vee p_2 \vee q_1} \tag{2.2}$$

The necessary condition for above is that $C$ should not be function of any of the $p_1, p_2, q_1$.

**Example 2.9** *Show by resolution that* $(p \Rightarrow q) \Rightarrow [(r \wedge p) \Rightarrow (r \wedge q)]$ *is a tautology:*

$$
\begin{aligned}
&= \neg(\neg p \vee q) \vee [\neg(r \wedge p) \vee (r \wedge q)] \\
&= (p \wedge \neg q) \vee [(\neg r \vee \neg p) \vee (r \wedge q)] \\
&= (p \wedge \neg q) \vee [((\neg r \vee \neg p) \vee r) \wedge ((\neg r \vee \neg p) \vee q)] \\
&= (p \wedge \neg q) \vee [(r \vee \neg r \vee \neg p) \wedge (q \vee \neg r \vee \neg p)] \\
&= (p \wedge \neg q) \vee [(q \vee \neg r \vee \neg p)] \\
&= (q \vee \neg r \vee \neg p \vee p) \wedge (q \vee \neg r \vee \neg p \vee \neg q) \\
&= T \wedge T \\
&= T. \ \square
\end{aligned}
$$