Unit 6 Knowledge Representation Using Logic – I

Structure:

6.1 Introduction

In the last unit, you have learnt the concept called Knowledge Representation. You have also studied the other aspects of knowledge representation like its history, issues and properties. You also learnt the meaning of Knowledge. In this unit you will be introduced to representing simple facts in logic and the role of logic in Artificial Intelligence. You will also learn in this unit the syntax and semantics for propositional logic.

A logic is a formal language for representing facts and properties of a world in a precise, unambiguous way. The use of symbolic logic to represent knowledge is not new in that it predates the modern computer by a number of decades. Even so, the application of logic as a practical means of representing and manipulating knowledge in a computer was not demonstrated until the early 1960s. Since that time, numerous systems have been implemented with varying degrees of success. Today, First Order Predicate Logic (FOPL) or Predicate Calculus as it is sometimes called, has assumed one of the most important roles in AI for the representation of knowledge.

Objectives:

After studying this unit, you should be able to

- explain representing simple facts in logic using propositional logic

- list the advantages of propositional logic
- explain the role of logic in Artificial Intelligence
- explain syntax and semantics for Propositional Logic

6.2 Representing Simple Facts in Logic

Under this we shall first explore the use of propositional logic as a way of representing the sort of world knowledge that an AI system might need. Then we shall explore Predicate logic (first order) or Predicate Calculus but before that let us see why we should use logic in AI:

Why Logic?

It is important for people working in AI to know logic for several reasons:

- **Theory**
  Logic has a sound mathematical foundation; things can be proved about it.
- **Applications**
  For certain classes of applications (e.g., proving correctness of programs) logic is the representation of choice.
- **Comparison with Other Methods**
  Other representation methods are often reducible to logic. Knowing logic helps in understanding other methods and may help prevent reinvention of old techniques.

6.2.1 Propositional logic

The simplest logic of all is the *Propositional logic.* Propositional logic is appealing because it is simple to deal with and a decision procedure for it exists. We can easily represent real-world facts as logical *propositions* written as *well-formed formulas (wff)* in propositional logic. Propositional logic deals with the determination of the *truth* of a sentence. An allowable sentence is called the **syntax** of proposition. A syntax or sentence holds various **propositional symbols**, where each symbol holds a proposition that can either be *true* or *false*. The names of the symbols can be anything from alphabets like a, b or c to symbols like α, β or γ to variable names like *IsOld*, and may hold meaning relative to their contexts in the concept. Although, two propositions are constant as per the syntax and have a fixed meaning. They are:

- True - proposition that is *always-true*.

- False - proposition that is *always-false*.

In mathematical terms, these values would pronounce more like this:

$$u \Leftarrow a \wedge b \wedge c$$

In simple terms this means that an object is an instance of a concept *u* if the conditions a, b and c hold true simultaneously. So, for instance, we are given with a concept that says, *one is eligible to drive a car if one has a license, a car and a knowledge of driving*. To state this in propositional calculus, we would have to state this in the following mathematical notation:

CanDriveCar ← **HasLicence** ^ **HasCar** ^ **KnowsHowToDrive**

The standard logic symbols we use in this Self Learning Material (SLM) are:

for all	∀
there exists	∃
implies	→
not	~
or	∨
and	∧
if and only if	↔

In addition, left and right parenthesis, left and right braces, and the period will be used as delimiters for punctuations. For example, to represent the compound sentence “It is raining and the wind is blowing’’ we could write (R ∧ *B)* where R stands for the proposition “It is raining’’ and S stands for the proposition “the wind is blowing.’’ If we write (R ∨ B) we mean “It is raining or the wind is blowing or both.’’

In the figure 6.1, some simple facts in propositional logic are shown. Using these propositions, we could, for example, conclude from the fact that it is raining the fact that it is not sunny. But very quickly we run up the limitations of propositional logic.

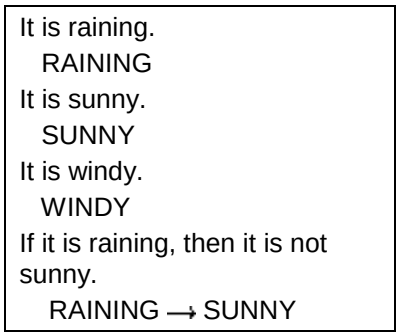

Figure 6.1: Some Simple Facts in Propositional Logic

Suppose we want to represent the obvious fact stated by the classical sentence

Socrates is a man.

We could write:

SOCRATESMAN

But if we also want to represent

Plato is a man.

We would have to write something such as:

PLATOMAN

which would be a totally separate assertion, and we would not be able to draw any conclusion about similarities between Socrates and Plato. It would be much better to represent these facts as:

MAN(SOCRATES)
MAN(PLATO)

Since now the structure of the representation reflects the structure of the knowledge itself. But to do that, we need to be able to use predicates applied to arguments. We are in even more difficulty if we are try to represent the equally classic sentence:

All men all mortal.

We could represent this as:

MORTALMAN

But this fails to capture the relationship between any individual being a man and that individual being a mortal. To do that, we really need variables and quantification unless we are willing to write separate statements about the mortality of every known man.

Hence we are forced to appear to move to predicate logic as a way of representing knowledge because it permits representations of things that can not reasonably be represented in propositional logic. In predicate logic, we can represent real-world facts as statements written as wff's.

Interpretation in propositional logic

An *interpretation* of a propositional logic formula is an assignment of a value (true or false) to each atom. There are 2^n possible interpretations of a formula with *n* atoms. Although this is large, it is finite; thus, every question about propositional logic is *decidable*.

Terminology:

- A formula is *valid* if it is true under every possible interpretation. Otherwise, it is *invalid*.
- A formula is *consistent* or *satisfiable* if it is true under some interpretation. If it is false under every interpretation, it is *inconsistent* or *unsatisfiable*.

  Clearly, a formula *G* is valid if and only if ¬*G* is inconsistent.

  If a formula *F* is true under an interpretation *I*, *I* is a *model* for *F*.

  Two formulas *F* and *G* are *equivalent* if they have the same values under every interpretation.

Advantages of propositional logic

- It is simple to deal with.
- There is a decision procedure for it.

6.2.2 First-order predicate logic

Propositional logic can only represent facts about the world. First-order logic describes a world which consists of objects and properties (or predicates) of those objects.

Among objects, various relations hold, e.g., Parent (Amit,Arti). A **function** is a relation in which there is only one value for a given input.

Examples:

Objects: people, houses, numbers, planets,...
Relations: parent, brother-of, greater-than,...
Properties: red, round, prime,...
Functions: father-of, one-more-than

First-order logic is universal in the sense that it can express anything that can be programmed.

Why first-order predicate logic?

Today, First Order Predicate Logic (FOPL) or Predicate Calculus has assumed one of the most important roles in AI for the representation of knowledge due to the following reasons:

First Logic offers the only formal approach to reasoning that has a sound theoretical foundation. This is especially important in our attempts to mechanize or automate the reasoning process in that inferences should be correct and logically sound.

Second The structure of FOPL is flexible enough to permit the accurate representation of natural language reasonably well. This too is important in AI systems since most knowledge must originate with and be consumed by humans. To be effective, transformations between natural language and any representation scheme must be natural and easy.

Third FOPL is widely accepted by workers in the AI field as one of the most useful representation methods. It is commonly used in program designs and widely discussed in literature. Understanding many of the AI articles and research papers requires a comprehensive knowledge of FOPL as well as some related logics.

Example of predicate logic (1):

Consider the following:

- Prince is a mega star.
- Mega stars are rich.
- Rich people have fast cars.
- Fast cars consume a lot of petrol.

and try to draw the conclusion: *Prince's car consumes a lot of petrol*.

So we can translate *Prince is a mega star* into: *mega_star(prince)* and *Mega stars are rich* int $\wedge$ o: $\forall$ *m: mega_star(m)* $\rightarrow$ *rich(m)*

Rich people have fast cars, the third axiom is more difficult:

- Is *cars* a relation and therefore *car(c,m)* says that case *c* is *m's* car. **OR**
- Is *cars* a function? So we may have *car_of(m)*.

Assume *cars* is a relation then axiom 3 may be written:

$\forall c,m: car(c,m)\ rich(m) \rightarrow fast(c)$

The fourth axiom is a general statement about *fast cars*. Let *consume(c)* mean that car *c* consumes a lot of petrol. Then we may write:

$\forall c: fast(c) \wedge m\exists: car(c,m) \rightarrow consume(c)$

Is this enough? NO! – Does prince have a car? We need the *car_of* function after all (and addition to *car*): $\forall$ *c:car(car_of(m),m)*. The result of applying *car_of* to *m* is *m*'s car. The final set of predicates is: *mega_star(prince)* $\forall$*m: mega_star(m)* $\rightarrow$ *rich(m)* $\forall$*c:car(car_of(m),m)*. $\forall$*c,m: car(c,m)* $\wedge$ *rich(m)* $\rightarrow$ *fast(c).* $\forall$*c: fast(c)* $\wedge$ $\exists$ *m: car(c,m)* $\rightarrow$ *consume(c).* Given this we could conclude: *consume(car_of(prince))*.

Example of predicate logic (2):

Consider the following set of sentences:

1. Marcus was a man.
2. Marcus was a Pompeian.
3. All Pompeians were Romans.
4. Caesar was a ruler.
5. All Romans were loyal to Caesar or hated him.
6. Everyone is loyal to someone.
7. People only try to assassinate rulers they are not loyal to.
8. Marcus tried to assassinate Caesar.

The facts described by these sentences can be represented as a set of wff's in predicate logic as follows:

- Marcus was a man.
  man(Marcus)
- Marcus was a Pompeian.
  Pompeian(Marcus)
- All Pompeians were Romans.
  $\forall$ x : Pompeian(x) $\rightarrow$ Roman(x)
- Caesar was a ruler.
  Ruler (Caesar)

Here we ignore the fact that proper names are often not references to unique individuals, since many people share the same name. Sometimes

deciding which of several people of the same name is being referred to in a particular statement may require a fair amount of knowledge and reasoning.

- All Romans were loyal to Caesar or hated him.

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

In English, the word “or” sometimes means the logical *inclusive-or* and some-times means the logical *exclusive-or* (XOR). Here we have used the inclusive interpretation.

Using exclusive-or same statement can be written as

$\forall x : \text{Roman}(x) \rightarrow [(\text{loyalto}(x, \text{Caesar}) \vee \text{hate}(x, \text{Caesar})) \wedge \neg(\text{loyalto}(x, \text{Caesar}) \wedge \text{hate}(x, \text{Caesar}))]$

- Everyone is loyal to someone.

  $\forall x : \exists y : \text{loyalto}(x,y)$

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

  $\forall x: \forall y: \text{person}(x) \wedge \text{ruler}(y) \wedge \text{tryassassinate}(x, y) \rightarrow \neg\text{loyalto}(x, y)$

- Marcus tried to assassinate Caesar.

  $\text{tryassassinate}(\text{Marcus}, \text{Caesar})$

From this brief attempt to convert English sentences into logical statements, it should be clear how difficult the task is.

Suppose we want to use these statements to answer the question

Was Marcus loyal to Caesar?

It seems that using 7 & 8, we should be able to prove that Marcus was not loyal to Caesar. Now let us try to produce a formal proof, reasoning backward from the desired goal:

$\neg\text{loyalto}(\text{Marcus}, \text{Caesar})$

In order to prove the goal, we need to use the rules of inference to transform it into another goal (or possibly a set of goals) that can in turn be transformed, and so on, until there are no unsatisfied goals remaining. This process may require the search of an AND-OR graph when there are alternative ways of satisfying individual goals. Here, for simplicity, we show only a single path. Figure 6.2 is an attempt to produce a proof of the goal by reducing the set of necessary but as yet unattained goals to the empty set. The attempt fails, however, since there is no way to satisfy the goal *person(Marcus)* with the statements we have available.

The problem is that, although we know that Marcus was a man, we do not have any way to conclude from that that Marcus was a person. We need to add the representation of another fact to our system, namely:

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

Now we can satisfy the last goal and produce a proof that Marcus was not loyal to Caesar.

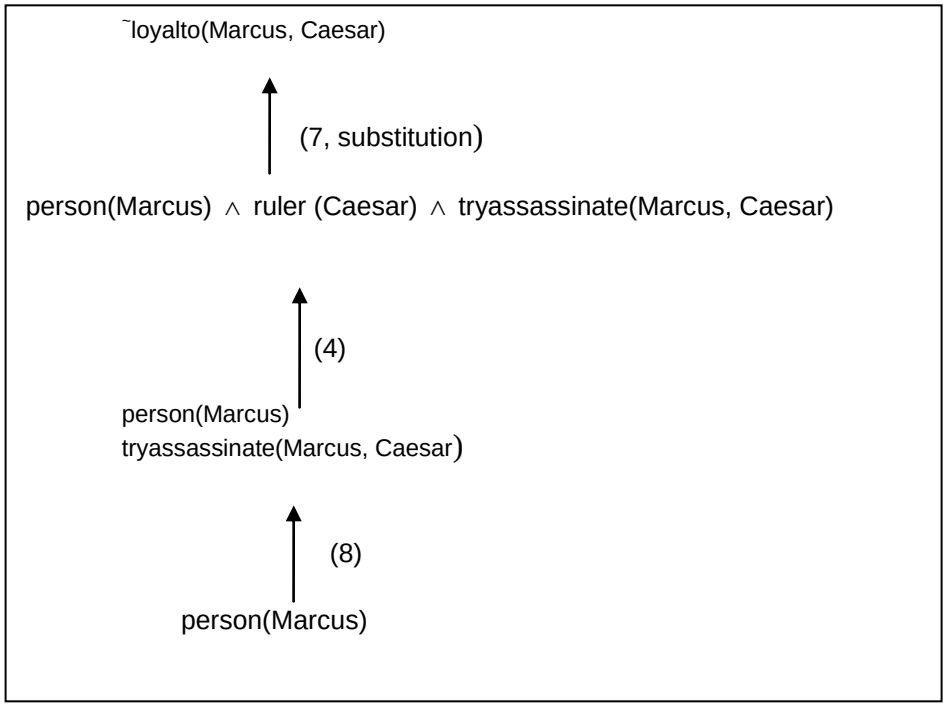

Figure 6.2: Proof of ¯loyalto(Marcus, Caesar)

Self Assessment Questions

1. FOPL stands for _____.
2. _____ logic deals with the determination of the *truth* of a sentence.
3. A _____ is a relation in which there is only one value for a given input.

6.3 Role of Logic in Artificial Intelligence

Theoretical computer science developed out of logic, the theory of computation (if this is to be considered a different subject from logic), and some related areas of mathematics. So theoretically minded computer

scientists are well informed about logic even when they aren't logicians. Computer scientists in general are familiar with the idea that logic provides techniques for analyzing the inferential properties of languages, and with the distinction between a high-level logical analysis of a reasoning problem and its implementations. Logic, for instance, can provide a specification for a programming language by characterizing a mapping from programs to the computations that they license. A compiler that implements the language can be incomplete, or even unsound, as long as in some sense it approximates the logical specification. This makes it possible for the involvement of logic in AI applications to vary from relatively weak uses in which the logic informs the implementation process with analytic insights, to strong uses in which the implementation algorithm can be shown to be sound and complete. In some cases, a working system is inspired by ideas from logic, but acquires features that at first seem logically problematic but can later be explained by developing new ideas in logical theory. This sort of thing has happened, for instance, in logic programming.

In particular, logical theories in AI are independent from implementations. They can be used to provide insights into the reasoning problem without directly informing the implementation. Direct implementations of ideas from logic – theorem-proving and model-construction techniques – are used in AI, but the AI theorists who rely on logic to model their problem areas are free to use other implementation techniques as well. Thus, we can distinguish three uses of logic in AI; as a tool of analysis, as a basis for knowledge representation, and as a programming language.

A large part of the effort of developing limited-objective reasoning systems goes into the management of large, complex bodies of declarative information. It is generally recognized in AI that it is important to treat the representation of this information, and the reasoning that goes along with it, as a separate task, with its own research problems.

The evolution of expert systems illustrates the point. The earliest expert systems, such as MYCIN (a program that reasons about bacterial infections) were based entirely on large systems of procedural rules, with no separate representation of the background knowledge – for instance, the taxonomy of the infectious organisms about which the system reasoned was not represented.

Later generation expert systems show a greater modularity in their design. A separate knowledge representation component is useful for software engineering purposes – it is much better to have a single representation of a general fact that can have many different uses, since this makes the system easier to develop and to modify. And this design turns out to be essential in enabling these systems to deliver explanations as well as mere conclusions.

Languages with precisely defined syntax and semantics are called **logics**. In a logic, we can develop inference mechanisms for an agent that uses the language. A sentence inside of a computer is not a fact, it is a representation of a fact. An inference mechanism must ensure that it allows new sentences to be derived from existing ones only when the new fact is true just as the existing ones are.

We want to generate sentences that are necessarily true given that the old sentences are true. This relationship between sentences is called entailment.

6.4 Syntax and Semantics for Propositional Logic

Valid statements or sentences are determined according to the rules of propositional syntax. This syntax governs the combination of basic building blocks such as propositions and logical connectives. Propositions are elementary atomic sentences. We can use the term formulas or well-formed formulas in place of sentences. Propositions may be either true or false but may take on no other value.

Some examples of simple propositions are:

It is raining.
My car is painted silver.
Amit and Arti have two children.
Snow is white.
People live on the moon.

Compound propositions are formed from atomic formulas using the logical connectives not and or if … then, and if and only if. For example, the following are compound formulas.

It is raining and the wind is blowing.
The moon is made of green cheese or it is not.
If you study hard you will be rewarded.

The sum of 10 and 20 is not 60.

Connection between Sentence and facts to be provided by semantics is shown in the figure 6.3.

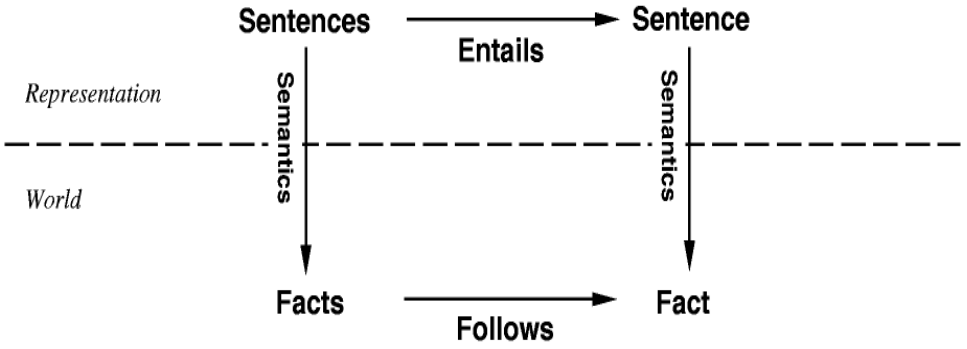

Figure 6.3: Connection between Sentence and facts to be provided by Semantics

6.4.1 Syntax

The symbols are True and False, propositional symbols such as P or Q, the logical connectives, $\wedge$, $\vee$, $\leftrightarrow$, $\rightarrow$, ~ and parentheses. All sentences are made by putting these symbols together using the following rules:

- The logical constants True and False are sentences.
- A propositional symbol by itself is a sentence.
- Wrapping a sentence in parentheses yields as sentence.
- A sentence can be formed by combining simpler sentences using the logical connectives.

If P and Q are formulas, the following are formulas:

($\neg$P)
(P $\vee$ Q)
(P $\wedge$ Q)
(P $\rightarrow$ Q)
(P $\leftrightarrow$ Q)

All formulas are generated from a finite number of the above operations.

An example of a compound formula is **((P $\wedge$ (~Q $\vee$ R)) $\rightarrow$ (Q$\rightarrow$ S))**

When there is no chance for ambiguity, we will omit parenthesis for brevity: **($\neg$(P $\wedge$ ($\neg$Q)))** can be written as **$\neg$(P $\wedge$ $\neg$ Q)**

Precedence of operators

Precedence of operators from highest to lowest is: ˜, ∧ , ∨ , →, ↔.

For example, to add parenthesis correctly to the sentence

P ∧ ˜ Q ∨ R → S ↔ U V W

We write

((((P ∧ ˜(Q)) ∨ R) → S) ↔ (U V W))

6.4.2 Semantics

The semantics or meaning of a sentence is just the value true or false; that is, it is an assignment of a truth value to the sentence. The values true and false should not be confused with the symbols T and F which can appear in a sentence. Note however, that they are not concerned here with philosophical issues related to meaning but only in determining the truthfulness or falsehood of formulas when a particular interpretation is given to its propositions. An *interpretation* for a sentence or group of sentences is an assignment of a truth value to each propositional symbol. As an example, consider the statement **(P ∧ ˜Q).** One interpretation (I_1) assigns true to P and false to Q. A different interpretation (I_2) assigns true to P and true to Q. Clearly there are four distinct interpretations for this sentence.

Once an interpretation has been given to a sentence, its truth value can be determined. This is done by repeated application of semantics rules to larger and larger parts of the statement until a single truth value is determined. The semantic rules are summarized in Table 6.1 where t and t' denote any true statements, f and f' denote any false statements, and a is any statement.

Table 6.1: Semantic Rules for Statements

Rule Number	True statements	False statements
1.	**T**	**F**
2.	**˜f**	**˜t**
3.	**t ∧ t'**	**f ∧ a**
4.	**t ∨ a**	**a ∧ f**
6.	**a ∨ t**	**f ∨ f'**
6.	**a → t**	**t → f**
7.	**f → a**	**t ↔f**
8.	**t ↔ *t'***	**f ↔ *t***
9.	**f ↔ *f'***	

We can now find the meaning of any statement given an interpretation I for the statement. For example, let I assign true to P, false to Q and false to R in the statement

$$((P \wedge \neg Q) \rightarrow R) \vee Q$$

Application of rule 2 then gives ¬Q as true, rule 3 gives (P ∧ ¬Q) as true, rule 6 gives (P ∧ ¬Q) → R as false, and rule 6 gives the statement value as false.

Some equivalence laws are shown in table 6.2.

Table 6.2: Equivalence laws

Idempotency	$P \vee P = P$
	$P \wedge P = P$
Associativity	$(P \vee Q) \vee R = P \vee (Q \vee R)$
	$(P \wedge Q) \wedge R = P \wedge (Q \wedge R)$
Commutativity	$P \vee Q = Q \vee P$
	$P \wedge Q = Q \wedge P$
	$P \leftrightarrow Q = Q \leftrightarrow P$
Distributivity	$P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$
	$P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$
De Morgan's laws	$\neg(P \vee Q) = \neg P \wedge \neg Q$
	$\neg(P \wedge Q) = \neg P \vee \neg Q$
Conditional élimination	$P \rightarrow Q = \neg P \vee Q$
Bi-conditional élimination	$P \leftrightarrow Q = (P \rightarrow Q) \wedge (Q \rightarrow P)$

Figure 6.4 shows the representation of some of the laws.

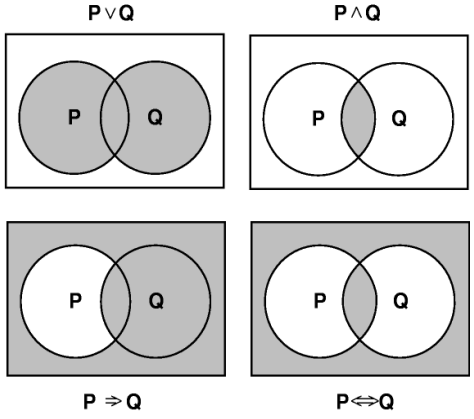

Figure 6.4: Representation of some of the laws

How to determine the equivalence of two sentences?

One way to determine the equivalence of two sentences is by using truth tables. For example, to show that $P \rightarrow Q$ is equivalent to $\neg P \vee Q$ and that $P \leftrightarrow Q$ is equivalent to the expression $(P \rightarrow Q) \wedge (Q \rightarrow P)$, a truth table such as the Table 6.3, can be constructed to verify or disprove the equivalences.

Table 6.3: Truth Table for Equivalent Sentences

P	Q	$\neg P$	$\neg(P \vee Q)$	$(P \rightarrow Q)$	$(Q \rightarrow P)$	$(P \rightarrow Q) \wedge (Q \rightarrow P)$
true	True	false	True	true	True	true
true	False	false	false	false	True	false
false	True	true	True	true	False	false
false	False	true	True	true	True	true

Self Assessment Questions

4. Languages with precisely defined syntax and semantics are called _____.
5. _____ are elementary atomic sentences.
6. An _____ for a sentence or group of sentences is an assignment of a truth value to each propositional symbol.
7. One way to determine the equivalence of two sentences is by using _____.

6.5 Summary

In this unit you have learnt to represent simple facts in logic, the role of logic in Artificial Intelligence. You have also learnt in this unit the syntax and semantics for propositional logic. Propositional logic deals with the determination of the truth of a sentence. An allowable sentence is called the syntax of proposition. FOPL is widely accepted by workers in the AI field as one of the most useful representation methods. Valid statements or sentences are determined according to the rules of propositional syntax. Later generation expert systems show a greater modularity in their design. Once an interpretation has been given to a sentence, its truth value can be determined. An interpretation for a sentence or group of sentences is an assignment of a truth value to each propositional symbol. Once an interpretation has been given to a sentence, its truth value can be

determined by repeated application of semantics rules to larger and larger parts of the statement until a single truth value is determined.

6.6 Terminal Questions

1. Describe Propositional Logic.
2. Explain First-Order Predicate Logic.
3. Explain the role of logic in Artificial Intelligence.
4. Write a note on semantics.

6.7 Answers

Self Assessment Questions

1. First Order Predicate Logic
2. Propositional
3. Function
4. Logics.
5. Propositions
6. Interpretation
7. truth tables.

Terminal Questions

1. Propositional logic is appealing because it is simple to deal with and a decision procedure for it exists.(Refer section 6.2 for detail)
2. First Order Predicate Logic (FOPL) describes a world which consists of objects and properties (or predicates) of those objects. (Refer sub-section 6.2.2)
3. A compiler that implements the language can be incomplete, or even unsound, as long as in some sense it approximates the logical specification. (Refer section 6.3.)
4. The semantics or meaning of a sentence is just the value true or false; that is, it is an assignment of a truth value to the sentence. (Refer sub-section 6.4.2)