UNIT 3

Knowledge and Reasoning

Learning Objectives

- To understand the importance of knowledge and its approaches
- To study the various representations of knowledge
- To study the role of logic in representation
- To understand the use of predicate and first order logic
- To study the inference and reasoning process
- To appreciate the relationship between knowledge and reasoning

INTRODUCTION

Search techniques and intelligent agents acquire information from environment and further build the knowledge with reference to problem at hand. It is this knowledge that is exploited further with the actions and decisions. Thus, appropriate and precise representation of the knowledge becomes a critical factor in the process. With the previous study on the search techniques and the applications, the agent environment, and the problem-solving issues, we now turn towards the knowledge representation. For all the methods that have been discussed previously, we need to look at the most important aspect of how the knowledge can be represented so that it can be used effectively and applied to the process. In turn, we can say that there is a reasoning process that actually is making the use of the knowledge.

But then, a question arises what is 'knowledge' basically? It is some set of patterns and associations derived from data or information that helps in making decisions and resolves problems that may be related to any day-to-day life or some complex problems. Consider a simple example, where a teacher has to judge the performance of a student for some exam. The teacher judges on the basis of percentile the student would obtain. The judgement could be based on the previous performances of that student or on some information given by some other teachers about that student. This can be considered as the available knowledge. So, one can arrive at some decisions based on this knowledge.

A systematic reasoning process is required when we try to relate the events to the outcomes or to arrive at judgements. Hence, *reasoning* is the way we conclude on different aspects of problems based on the available knowledge representation. Logic plays an important role in the reasoning process. So, *logic* is the one that makes the knowledge representative. In the course of representing the knowledge and utilising it, the chapter focuses on the various aspects that are essential from appropriate representations to exploit the knowledge base (KB). Though there are restrictions with the knowledge base handled here, it will be a stepping stone for us to start with the representations and understanding its use, where the knowledge is certain.

Let us begin our discussion with the knowledge representation, various issues and other aspects related to knowledge representation along with the agent environment.

7.1 KNOWLEDGE REPRESENTATION

In introduction, we have mentioned that knowledge is an important aspect of the reasoning process. Our outcomes govern the way we have the knowledge and the way we update it. Mapping it in technical terms, depending on the domain one is working on, it is very crucial to identify and create representation of the knowledge. So, can we say that knowledge representation is about representation of the facts at hand? The answer truly lies in the fact, viz. which facts at hand can be represented or to be specific, which can be manipulated. This is required from the viewpoint of specific knowledge representation that gives a broader view for problem solving. Let us proceed with the approaches and issues of knowledge representation.

7.1.1 Approaches and Issues of Knowledge Representation

Consider a case where fruits are to be arranged in a basket. This would be a simple task at present. One can easily have a knowledge representation and act accordingly. There could be multiple options that will give the desired outcome. But with the addition of more facts like the size of the fruits, the quantity, the basket size and so on, it would further narrow down the arrangements. Further, it is quiet obvious that some information gets available

over a period of time. The KB representation should be able to handle this sort of environment, where limited or partial information is available in the beginning and later on, more information is evolved. We cannot say that for a particular problem, there is only a specific way of knowledge representation. But the representation definitely counts while coming to reasoning. Figure 7.1 depicts the process of deriving facts. From the diagram, it is clear that raw data, domain knowledge and, percept allow to refine internal representation iteratively to arrive at the final facts. Here, reasoning is required to establish relationships among the available data and the final facts.

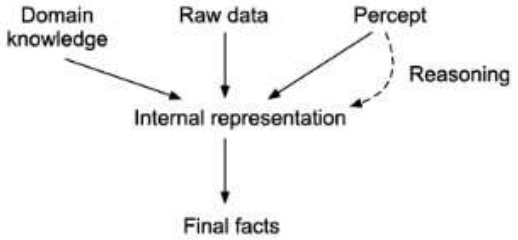

Figure 7.1 Knowledge building and representation.

Approaches

Before we start our discussion on the approaches, let us have a quick look on the properties that are required for knowledge representation in the system design. While designing a system for knowledge representation, we would always go for a system that allows representation of the entire knowledge, which is required with respect to the application or domain we work on. This is precisely the property that speaks about adequacy in terms of representation. The next property talks about adequacy in terms of inferring. Here, it is necessary that the knowledge should be represented in such a way so that there is a way to manipulate the representative data in order to derive at new ones. At the same time, it needs to be efficient in terms of inferring, where the additional data should be used in the direction of better inferring. Further, we would like to have the representation to learn; it should be adaptive and able to accommodate new additional information that would be available at any point of time. This is the property of efficiency in term of acquisition. Here, we mean to say that it should posses the property of being incremental. This could also be a simple knowledge update by the user himself.

During the course of representation, it is worth to mention that even though we expect it to possess the properties, it is not feasible that just one representation is able to fulfil them. Generally, different approaches are in use when it comes to representation of knowledge, even though it is for a specific domain.

Let us start with the basic and the simplest ways for representation. Now, when we are discussing about the representation, the obvious thing that comes to our mind is the database or files. Are they a part of knowledge representation? Are there some other methods too? Definitely, they are a part of representation. Let us understand what approaches can exist along with them with respect to the structure.

We begin with a simple *relational knowledge structure*, wherein we have the database representation. The facts can be mapped into the relations and stored in the database. This kind of representation without any additional procedure to get something out of it is a weak inference mechanism.

Table 7.1 represents a simple example of relational knowledge structure.

TABLE 7.1 RELATIONAL KNOWLEDGE STRUCTURE

Employee	*Salary*	*Experience*
Sameer	30000	3
Kavita	20000	2
Jasmin	20000	2

The next structure is *inheritable knowledge structure*. This type of representation is required as all the knowledge related to the inheritance is not mapped in the earlier case. This is critical while drawing conclusions. Hence, we need a structure that can help in inferring. So, a general hierarchy structure is used, where it is possible to have a proper mapping with inheritance. Figure 7.2 shows this type of knowledge structure.

Let us take the example of cricket. The knowledge structure for any player is based on various parameters or attributes. The player can be a batsman or a bowler. He can be a right-handed player, with some specific height. So, there exists an ISA relationship among a person, a player and a batsman or a bowler. Similarly, we can have an instance to represent the knowledge. The structure is also called *slot-filler structure*.

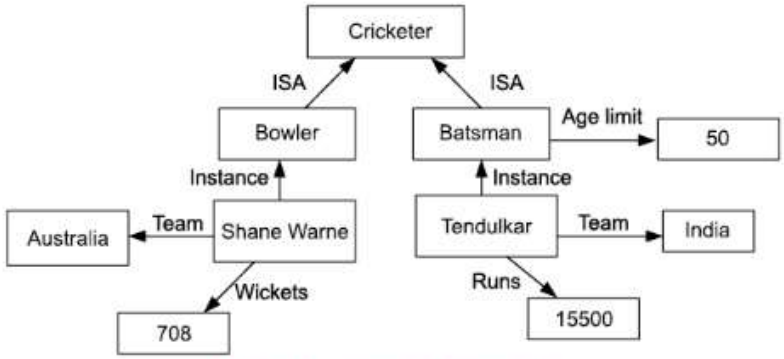

Figure 7.2 Inheritable knowledge.

Semantic network and frame collection are also related to the structure, which are discussed later in the chapter.

We are aware that finally our intention is to come out with the outcomes that will answer our queries. Now, for example, we say that we want to have a solution to a query like does Tendulkar satisfy age limit? (The diagram represents hypothetical data to understand the hierarchy). The process of seeking a solution is

1. To check if direct answer is available. In case, it is not, go to the next step
2. To check for attribute instance

Here, it will be under the category of Batsman. Had it been the case that the query was regarding the parameter like related to cricketer, we would have to move a step up in the hierarchy and got the answer. This process or the approach applies the property of inheritance. This approach actually guides the utilisation of the knowledge.

Another structure is *inferential knowledge structure*. In this type of structure, first order predicate logic is used. So, a typical example is combining the knowledge to get the outcomes. (Details of predicate logic are discussed later in the chapter.) One point to mention here is that there is a need for the inference procedure to have the utilisation of knowledge.

Procedural knowledge structure is another approach for the representation. This structure comes into picture when we need to have the knowledge in detailed form. Somewhere, it specifies steps to be followed and its details. Programming languages are used for that purpose. LISP is the most common language that finds a place in this case. Another representative structure could be in the form of rules—production rules.

We have now discussed the introduction to structures. As we will proceed through the chapter, we will have a clear picture of the other details about the knowledge representation and use of logic for the same. Let us turn towards the issues of knowledge representation.

Issues of Knowledge Representation

There are various issues of knowledge representation related to information mapping and use of structures. Most obvious questions that arise in knowledge representation are as follows:

1. Which information can be mapped into knowledge?
2. How can it be mapped?
3. How to decide which would be an ideal mapping that will give the most accurate solution to the reasoning process?
4. Is there a way that will help in better representation?
5. What would be the memory constraints?
6. Is it possible to have an access to the relevant part of the knowledge? There could be many more questions also.

Since we have already studied representative structures, we will focus on the issues related to knowledge representation and methods to resolve the same.

1. *Attributes* are the most important ones that have an impact on the representation. It is required to understand and identify the important attributes. This helps in absorbing the important parameters for the KB representation.
2. Similarly, identifying the *relationships among the attributes* in the representation is equally important. By pairing of the attributes or with ISA methods, the relationships are captured. Reasoning about values that are taken up also adds up complexities in the selection process. Hence, proper selection of the attributes impacts the relationships.
3. Handling the issue of level or upto what depth the mapping of the knowledge is to be done defines the *granularity*. This is governed by the availability of facts and the level upto which it is possible to split them and represent them. Sometimes, splitting can prove to be an option to handle the issue, making the easy accessibility, but sometimes, it adds complexities to handle the data.
4. Further, the issue is *representation of the objects as sets*. Use of logic and the inheritance that we have already studied are very well-suited to tackle this issue. Sometimes, a particular property that is applicable to the entire set has an exception. There is a need to clarify in terms of properties before we have the representations. The name that represents the set has to be precise, as this impacts the object selection, letting us know about its membership. Here, extensional definition is used, where the members are listed in the set, and the other is intensional, where rules are provided that determine the belongingness of the object. This tries to restrict the representation of the objects.
5. Finally, *selection of correct structure* is the most important part to have a proper representation. Things, here, as made clear earlier too, are dependent on the domain. Here, the issues range from how to fill in the details to when to have a new structure. Though selecting a knowledge structure that matches a particular problem is very difficult, still to mention a few, one method can be indexing the structure with significant keywords. This can with regard to English, but then this is specific when the domain description is available in English. Other approach could be the use of pointers. Using this notion, it is possible to have intersections of the sets and to use structure that are precise. One more option could be locating and selecting a major clue in the problem. This helps in defining the structure. It is also equally important that the structure is flexible, which allows revisions as per the requirements. Getting appropriate values is the final requirement. In the

sense, whenever we have structure selection, its exploitation is done by getting the results. Generally, a candidate structure is set up and then applied to the problem. But, if the outcomes are not accurate for the problem, then it is the time to change/revise the structure. This can be in terms of attribute values too.

Figure 7.3 represents the issues and their inter-relationships.

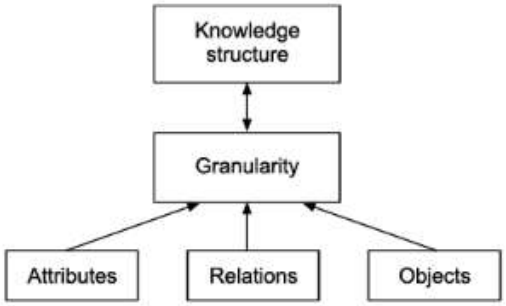

Figure 7.3 Knowledge structure hierarchy.

7.2 KNOWLEDGE-BASED AGENTS

We have already studied about intelligent agent. An *agent* is the one who acts according to the environment. To act, it needs to know the knowledge of how to act. A knowledge base (KB), hence, plays an important role in deciding the actions. This knowledge base is nothing but a representation of the information that helps an agent to act. Knowledge representation language makes use of sentences to represent facts about the world.

An agent's operating environment is based on perception and action. Depending on the current percept, it acts. So, this percept should be available with the knowledge base. Then, the action occurs. This action too needs to be updated to the knowledge base. We can say that the knowledge-based agent performs three-fold task. A typical algorithm is summarised below:

Knowledge-based agent
Input: Percept
Returns: Action
  Give-info (KB, make-percept-sentence (p, t))
  Action ← (KB, make-query(t))
  Give-info (KB, make-action-sentence (a, t))

where p is the current percept, t is the time and a is the action.

From the steps, it is clear that in the initial *give-info* function, it records the current percept p and reveals the information about the percept in the form of knowledge. This can be in the form of a sentence. In the next step, the action a is returned that is the result of formulation of query for the knowledge that has been received at the previous stage. The last stage is where the KB is updated about the action taken.

The basic difference between a normal agent and a knowledge-based agent is that the actions of knowledge-based agent are not arbitrary. They are dependent on the knowledge. They are described at three levels. At the knowledge level, the agent is provided with the information it should know and its targets. At the logical level knowledge is represented in logical language, while at implementation level, logical sentences are implemented. The implementation level does not hamper any of the knowledge level details.

Knowledge-based agent accepts new tasks through defined goals. They acquire knowledge through learning and flexibility adapts to situations. Before the agent starts its operation, the initial knowledge can be made available in two ways: (i) declarative and (ii) procedural.

Declarative knowledge is embedded in the system in the form of pre-defined rules, while procedural infers about the action with reference to situations.

7.3 THE WUMPUS WORLD

This is a very traditional environment to understand an intelligent agent-based system. The Wumpus World is basically a cave that has some rooms connected to each other by passways. In one of the rooms lies a wumpus (a beast) that eats anyone who enters the room. The wumpus can be shot by an agent, but the agent has only one arrow available. In some of the rooms, there are pits too. The best part of this world is that the agent might get a heap of gold. The PEAS description for this environment is as follows:

P—Performance: (i) 1000 points when gold is found
(ii) −1000 points when falls in pit
(iii) −1 for every move
(iv) −10 when arrow is used

E—Environment: A grid of 4 * 4, with pits at some squares and gold at one square and agent position at [1, 1] facing right.

A—Actuators: (i) Turn 90° left/right
(ii) Walk one square forward
(iii) Grab or take an object
(iv) Shoot the arrow (agent has one arrow)

S—Sensors: There are five sensors. They capture the following:

1. In rooms adjacent to room of wumpus (excluding diagonal), the agent perceives stench.

2. In the square adjacent to pit (excluding diagonal), agent the perceives breeze.
3. In the room containing gold, the agent perceives glitter.
4. When agent walks in a wall, he perceives a bump.
5. When wumpus is killed, it screams that can be perceived anywhere in the environment.

Figure 7.4 shows the Wumpus World. (Different positions can exist for the wumpus, pits and gold and it is not the case that the room arrangement is same).

The agent in the Wumpus World draws conclusions based of the facts. If the facts are correct, naturally the conclusions will be correct and hence, its actions too. Let us understand the actions of agent in the environment.

The knowledge base of the agent contains five initial conditions that are listed in the sensors. It starts with [1,1]. It knows that this is the position, where it is safe. It needs to move ahead either to reach the room of gold or to be safe. Since at [1,1], it does not perceive a stench or breeze, the adjacent rooms are safe as concluded based on the facts. So, it can move to [2,1] or [1,2]. Let us say it goes to [2,1]. At this point, it gets a breeze. So, it concludes that at [2,2], there could be a pit or at [3,1], there could be a pit.

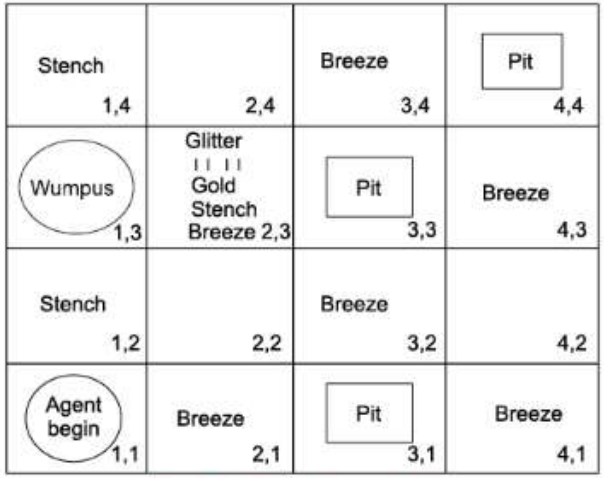

Figure 7.4 The Wumpus World.

So, it moves back. It now goes to [1,2]. It perceives a stench. This indicates the there is wumpus. The wumpus could be at [1,3] or [2,2]. But [2,2] is the position, where the agent assumes that there could be a pit. Since it does not receive a breeze in [1,2], it concludes that pit is absent in [2,2]. But then what about the wumpus? If the wumpus was present in [2,2], it would detect a stench at [2,1]. Hence, it concludes that [2,2] is safe and the wumpus is present at [1,3]. Concluding this is difficult without previous experiences. From here, the agent could go to [2,3] or [3,2]. If it selects [2,3], he will perceive glitter!

Hence, it is the reasoning that leads the agent to take correct actions and this is dependent on the correctness of the facts. So, we conclude that logical reasoning is the essence to have correct conclusions.

7.4 LOGIC

What are we actually doing by studying logic? Are we trying to understand the process of reasoning? We already know that logic basically deals with the study of principles of reasoning. So, what we are trying to put forth in this section is how logic is built, or rather how the logical representations help in the process of decisions.

So, can we say that there are syntax and semantics that are required to be handled in logical representations? Yes. Logic involves syntax, semantics as well as inference procedure. Speaking about the syntax, we know that there can be a variety of ways to represent it. It is not a concern, actually, but it is the way an agent (relating it to the Wumpus World's agent) builds the base so that the reasoning plays an important part. With respect to semantics, though it deals with the meaning, in sentential form, it can either be true or false. There is a need to define logical involvement in two sentences or facts. So, here, we are trying to model it. It can be put forth in the following way: Assume that x and y are the two sentences whose relationship is to be determined. Let us map this relationship as follows:

$$x \models y$$

This indicates that the sentence x entails y, indicating that in the model when x is true, y is also true. For example, suppose

$$KB = \{p=1\}$$

$$\text{Then, } KB \models \{p + 1 = 2\}$$

The other way round, it states that y is contained in x. In the Wumpus World, when the percepts are combined with the facts or the rules, then the combination constitutes the Knowledge base (KB). Considering the example of Wumpus World, where the agent wants to infer whether a pit exists in the rooms, say [1,2], [2,2] and [3,1] with available information that breeze exists at [2,1] and at [1,1], it experiences nothing, then there are 2^3 possible combinations. With the available facts, the KB is definitely false in the case, where the agent wants to infer about the pit in [1,2]. This is because it does not experience any breeze in [1,1]. To infer, two cases are considered—1. Pit does not exist in [1,2]
and 2. Pit does not exist in [2,2]. From the possible models shown in Figure 7.5, it can be inferred that the KB |= case 1. So, when every KB is true, case 1 will be true. But it

cannot be judged as to whether the pit exists in [2,2]. This type of inferring is called *model checking*.

So, in logical inferring, there is a notion of truth that is to be maintained. Even it needs to have the property of completeness. Finally, the last word is if the knowledge base is true, then the sentence it derives has to be true.

7.5 PROPOSITIONAL LOGIC

After having an overview of logic part, we begin with the propositional logic; the most simple logic. Why are we studying it? The answer is—for reasoning. It is a mathematical model that provides reasoning regarding the logical value of an expression.

What is propositional logic? It is a logic that is concerned with the propositions and their relationships. Propositional logic is also called *sentential logic*. Propositions are sentences—declarative sentences, say 'Ice is cold' or 'The door is closed', and so on. But 'Is it cold?' or 'Open the door' are not propositions. Hence, they cannot be an explicit order. A declarative sentence states that it can be either true or false, but not both. Propositional logic is the fundamental logic. Let us understand the syntax and semantics first.

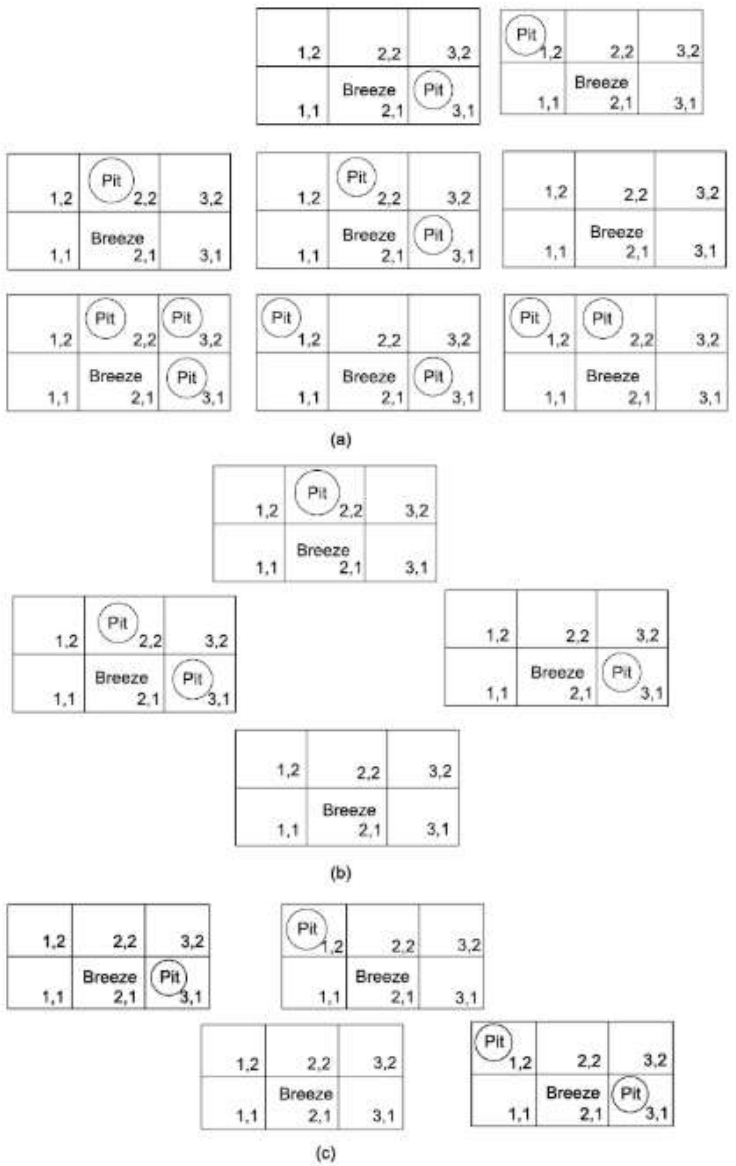

Figure 7.5 Possible (subset) cases in the Wumpus World for pits: (a) Knowledge base (KB), (b) Case 1: Subset representations, (c) Case 2: Subset representations.

7.5.1 Syntax, Semantics and Knowledge base Building

Syntax

In propositional logic, there exist two types of sentences—Simple and compound. An atomic sentence or simple sentence consists of a single propositional symbol. This symbol essentially represents if the proposition can be true or false. Does that mean that there are two propositional symbols with fixed meanings for true and false? The answer is yes. But as we study ahead in detail, things will be more clear.

The syntax of propositional logic basically defines the allowable sentences. As stated earlier, it is represented as symbols, they could be, say P, Q, L, M, and so on. A simple assertion is represented as p. This indicates that the proposition is true.

The five operators it has are briefed below:

1. Negation (~ or ¬)
2. Conjunction (∧)
3. Disjunction (∨)
4. Implication (→): If ....then
5. Biconditional (↔): iff- if and only if

In propositional logic, we need to be very specific with respect to syntax. The rule followed for precedence is highest from ¬ to lowest ↔. It is also essential that we make use of parenthesis for setting the precedence.

Semantics

Semantics tells about the rules to determine the truth of a sentence. Things are simple when it comes to simple sentence. But with the compound one, the model comprising a number of propositions actually defines the truth value. There can be many models depending on the values of the propositions considering their permutations. So, with 2 propositional symbols playing role, the models will be 2^2, whereas with 3, it will be 2^3. For the operators, let us define truth tables by which we will get clarity of the values they would take up.

Let P and Q be the propositions. The truth tables for the operators defined in earlier section are shown in Table 7.2. The final value, as we can see, is dependent on the values of P and Q.

TABLE 7.2 TRUTH TABLES

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$Q \rightarrow P$	$P \leftrightarrow Q$
False	False	True	False	False	True	True	True
False	True	True	False	True	True	False	False
True	False	False	False	True	False	True	False
True	True	False	True	True	True	True	True

From Table 7.2 we can see that the value of $P \leftrightarrow Q$ is true only when $Q \rightarrow P$ and $P \rightarrow Q$ are true. Consider the scenario of the Wumpus World again. There is a breeze in [1,1], let us represent this as B. Then, we know that a pit exists in [2,1] or [1,2]. Let P represents pit in room [2,1] and Q represents pit in room [1,2]. Mapping it in propositional logic, we represent it as follows:

$$B \leftrightarrow (P \vee Q)$$

But we cannot have the implication operator, as it results into incompleteness of knowledge.

Building a Knowledge Base (KB)

For a subset of the Wumpus World, a knowledge base is required to be built.

To build any KB, the first step is to decide the propositions. We then construct the rules with the operators and the true/false values of propositions. Consider that i and j represent the room or the grid value. Let $P_{[i,j]}$ represents the proposition that is true if there is a pit in i, j. Similarly, let $B_{[i,j]}$ represents the proposition that is true when there is a breeze in i, j.

The KB comprises the rules now that are built with the propositions and operators. As an example one rule can be as follows:

Rule 1: $\neg P_{[1,1]}$: This rule states that there is no pit in [1,1].

Similarly, the earlier rule that we have derived in the semantics is also a KB-representing rule. We can rewrite the rule 2 as follows:

Rule 2: $B_{[1,1]} \leftrightarrow P_{[1,2]} \vee P_{[2,1]}$. In this way we can define the rules forming our KB for the Wumpus World.

Finally, the KB comprises all the rules or the sentences in the conjunction form, i.e., Rule 1 ∧ Rule 2 ... ∧ Rule n. This, in turn, tells that all the rules are valid and true.

7.5.2 Inference

Now, once we have our KB represented, it is the time that we decide upon the inference. With the propositions that we have discussed considering the Wumpus World, they can have different models depending on the values. By models, as said earlier, we mean the different values which the propositions take in the compound statement. So, by inferring we mean that it should be decided whether KB $\models x$. Here, x is some sentence.

So, when it comes to inference, we need to enumerate the models. Now, while doing

so, the number of propositions playing part in it directly hamper the efficiency. The values where KB is true, and thus, the value of x is also true indicate inferring. As an example, assume that we have KB true for some model. Suppose you have a truth table of the values of propositions, rules and KB. A snapshot of the same is given below in Table 7.3.

TABLE 7.3 VALUES OF PROPOSITIONS—A SNAPSHOT

$B_{[1,1]}$	$B_{[2,1]}$	$P_{[1,2]}$	...	$P_{[3,1]}$	$B_{[2,1]}$	*Rule* 1	*Rule* 2	...	*KB*
⋮									
False	True	False		True		True	True		True
False	True	False		False		True	True		True
False	True	False		True		True	True		True
⋮									

From Table 7.3, we can say that ~($P_{[1,2]}$) is true, so we can infer that there is no pit in [1,2]. But the same cannot be true in case of $P_{[3,1]}$, as in one case, it is false. This is just a simple case considered to understand inferring, but just image the number of models that would be generated with the increase in the propositions.

Given a sentence n, the approach for deciding entailment is based on recursive enumeration. The approach is:

Given—KB, x, list of symbols in KB and x.

Check—1. If symbols are empty then
check if model is consistent with KB
if true then check if x evaluates to true
else it is inconsistent
2. Else
recursively construct conjunction
for partial models with symbols in KB and x.

Some concepts: Tautology, Contradiction and satisfiability

1. *Tautology:* A tautology states that the sentence is true if it is true in all models. That means a proposition is always true. For example ($P \vee \sim P$) will have true value, irrespective of the P values. Hence, it is a tautology, which is sometimes also called *validity*.

2. *Contradiction:* In contradiction, the proposition is always false. For example, $P \wedge \sim P$ will be always false, irrespective of the values taken by proposition P.

3. *Satisfiability:* A sentence or a proposition is satisfiable if it is true for some model. Let us say a sentence x is true in a model m then m satisfies x. m is a model of x.

Refutation

After studying satisfiablity, the question is how can it be determined? Can we say that we need to explore all the models and check them till we get one that satisfies the sentence? The answer is yes. To determine the satisfiability of a sentence is NP-complete problem. Now, when we talk about satisfiability, we can say that most of the problems related to computer science are satisfiable. (Constraint satisfaction and search problems have satisfiability playing a role in solving them).

Understanding the relation between the validity and satisfiability can be explained as follows:

Assume that a sentence x is valid. To prove that x is valid, we need to prove that $\sim x$ is satisfiable. Other way round, x is satisfiable iff $\sim x$ is not valid. Referring to KB representations, we can state that

$x \models y$ iff sentence $(x \wedge \sim y)$ is unsatisfiable.

Proving y from x by checking the unsatisfiability as mentioned above is *proof by refutation* or *contradiction*. This relates to the mathematical solving of problems, where by treating a value as false, we move into contradictions, and hence, carry the proof of refutation.

7.5.3 Reasoning Patterns in Propositional Logic

In reasoning patterns, we use and apply the basic rules in order to derive chains of conclusions to get the outcome or the target. These basic rules are also called *patterns of inference*.

The two most commonly used rules are *modus ponens* and *and-elimination*. Modus ponens is represented as follows:

$$\alpha \rightarrow \beta, \alpha \mapsto \beta$$

The rule states that when any sentence is in the form of $\alpha \rightarrow \beta$, and α is given, then we can infer β.

As an example, when we have the rule that (wumpus-ahead $\wedge$ wumpus-alive) $\rightarrow$ shoot and (wumpus-ahead $\wedge$ wumpus-alive); we can infer shoot. Considering one more example (mapping in propositional logic), suppose we have the following rule:

R: $\sim S_{[1,1]} \rightarrow \sim W_{[1,1]} \wedge \sim W_{[1,2]} \wedge \sim W_{[2,1]}$ and given that a stench is ahead

Then, with modus ponens, we get

$\sim W_{[1,1]} \wedge \sim W_{[1,2]} \wedge \sim W_{[2,1]}$, where β is inferred

In case of and-elimination, the rule is represented as follows:

$$\alpha_1 \wedge \alpha_2 \wedge \alpha_3 \wedge \alpha_n \mapsto \alpha_i$$

The rule states that from conjunctions, it is possible to infer any conjunction.

This also states that if we have wumpus-ahead ∧ wumpus-alive, then we can infer wumpus-alive. From the above inference with stench, we get simply the and-elimination inference as follows:

$$\sim W_{[1,1]} \sim W_{[1,2]} \sim W_{[2,1]}$$

Now, the next question is again what is the reason behind considering these rules? These rules are actually the ones that eliminate the need for generating the models. When these rules are applied, they generate sound inferences.

Resolution

The previous point clarifies the rules being sound. We now need to discuss the completeness. Any search algorithm (we have discussed about completeness in Chapter 3) is said to be complete when it is guaranteed to go to a reachable goal. But what if the rules available are insufficient or inadequate? Can we reach the goal?

Resolution is a single inference rule that is discussed in this section, which gives a complete inference algorithm when coupled with complete search algorithm.

Coming back to the Wumpus World's example, some rules are considered here to understand the resolution. Consider the case where the agent is in [1,2], a case where he has returned from [2,1] to [1,1] and then has gone to [1,2]. Here, he perceives a stench, but no breeze. Now, we add some more rules.

Rule a: $\sim B_{[1,2]}$

Rule b: $B_{[1,2]} \leftrightarrow (P_{[1,1]} \vee P_{[2,2]} \vee P_{[1,3]})$

Rule c: $\sim(P_{[2,2]})$

Rule d: $\sim(P_{[1,3]})$

Rules c and d imply that pit is not present in [2,2] and [1,3]. Similarly, we can derive that there can exist a pit in [1,1] or [2,2] or [3,1]. Rule *e* represents the same as follows:

Rule e: $P_{[1,1]} \vee P_{[2,2]} \vee P_{[3,1]}$

But where is the application of resolution and how do we apply it? The resolution rule is applied in the rules c and e. By this we mean to say that $\sim P_{[2,2]}$ in rule c resolves with $P_{[2,2]}$ in rule e. By applying this, we get

Rule f: $P_{[1,1]} \vee P_{[3,1]}$

But the initial rule that we have in KB building is that $\sim P_{[1,1]}$. This again resolves with the rule f. Hence, what we get is

Rule g: $P_{[3,1]}$

This inference rule states that if there is a pit in [1,1] or [3,1] (by rule f) and there is no pit in [1,1] (by resolution), then there is definitely a pit in [3,1].

The steps applied to infer the rules f and g are called *unit resolution inference rules*.

The propositions involved in the process are also called *literals*. Resolution is actually the basis for complete inference procedures. Any search algorithm that is complete and applies the resolution rule can derive any conclusion that is entailed in KB. Suppose we know that a proposition X is true. It is not possible to have a resolution to generate $X \vee Y$, but we can determine if $X \vee Y$ is true or not. This is referred to as *refutation completeness*.

Conjunctive normal form (CNF): From the rules that are mentioned in the resolution, it is noticed that the resolution is applied only to disjunctions. But then, the rules can be in conjunction form too. What we need to do is to convert them into CNF, which is a conjunction of disjunctions. CNF is conjunction of clauses, where clauses are disjunctions of literals.

While converting, the following steps are to be carried out:

1. If a bidirectional implication exists, it has to be eliminated in following way: $\alpha \leftrightarrow \beta$, is replaced with $\alpha \rightarrow \beta, \beta \rightarrow \alpha$.
2. If an implication exists, then it has to be eliminated in the following way: $\alpha \rightarrow \beta$ is replaced with $\neg \alpha \vee \beta$.
3. Use equivalence to have ~ or $\neg$.

By applying the DeMorgan's theorem, we can rewrite the rules as follows:

(i) $\neg(\neg \alpha) \equiv \alpha$

(ii) $\neg(\alpha \wedge \beta) \equiv (\neg \alpha \vee \neg \beta)$

(iii) $\neg(\alpha \vee \beta) \equiv (\neg \alpha \wedge \neg \beta)$

Once these rules are applied, we can have the resolution inference rule applied.

Let us take a simple example. Assume α and β are in CNF. Therefore, $\alpha \wedge \beta$ is also in CNF.

Now, conversion of $\alpha \vee \beta$ can be carried out as follows:

1. In a given scenario, if α and β are literals, then

$$\alpha \vee \beta \text{ is in CNF}$$

2. If $\alpha = \alpha_1 \wedge \alpha_2 \ldots \wedge \alpha_1$, then

$$\alpha \vee \beta = (\alpha_1 \vee \beta) \wedge (\alpha_2 \vee \beta) \ldots \wedge (\alpha_1 \vee \beta)$$

Assuming β is literal, then

$$(\alpha_1 \vee \beta) \wedge (\alpha_2 \vee \beta) \ldots \wedge (\alpha_1 \vee \beta) \text{ is in CNF}$$

Otherwise β is conjunction of β_1 to β_k and using distribution, we can convert each α_i and β_i to CNF.

Resolution algorithm: After having studied *refutation* in the previous section, we move ahead with the resolution algorithm. In refutation, proof is with contractions. For example, to prove that KB |= *y* we need to show (KB ∧ ~*y*) is unsatisfiable. While applying the resolution algorithm, we

1. Convert (KB ∧ ~*y*) into CNF.
2. We get some resulting clauses.
3. The resolution rule is to be applied to each clause.
4. The complementing pairs, are resolved to generate a new rule or a clause.
5. Add this to the KB if not already present.
6. Goto step 3 till any of the following two conditions occur:
   (i) No new clauses can be added in which KB does not entail α.
   (ii) Applying the resolution yields an empty rule, indicating KB entails α.

Forward and Backward Chaining

Resolution put forth an inference mechanism that is complete. But practically speaking, in most of the cases, we do not need to have the resolution. This is because the KB comprises restricted clauses or rules (clauses where at most a single proposition or literal is positive). A clause that is disjunction of literals of which at most one is positive is a *Horn clause*.

For example, the clause has ¬ $S_{[1,1]}$ ∨ ¬ Breeze ∨ $B_{[1,1]}$ is a Horn clause (¬$S_{[1,1]}$ is the state of the agent at 1,1). But a clause like ¬$B_{[1,4]}$ ∨ $P_{[2,1]}$ ∨ $P_{[1,2]}$ is not a Horn clause. The Horn clause containing exactly one positive literal is called *definite clause*. This literal is called *head* and other literals are said to be *body*. When there is not any negative proposition in a definite clause, it is called *fact*. A clause (for example, ¬$P_{[3,1]}$ ∨ ¬$P_{[1,3]}$ is same as writing $P_{[3,1]}$ ∧ $P_{[1,3]}$) is called *integrity constraint*. It points out the errors.

Forward chaining and backward chaining are the algorithms that are used for inferring. While applying the algorithm, the knowledge base comprises the Horn clauses. What are we trying to infer in these algorithms? Let us assume we have a query that is required to be resolved. This can be resolved by going through the KB to get the answer. This query is a proposition, say whether any position in the room has a pit or not. So, we are trying to find whether this proposition is entailed in the knowledge base.

Forward chaining: In forward chaining, the process starts with the known facts. From the known facts, it adds to the conclusion. This process is carried out till we reach out the query that we have to resolve or till no inference can occur further. The approach is mapped to an AND-OR graph. Here, the constraint is that till all the conjuncts are known, it does not proceed ahead.

The steps can be summarised as follows:

1. Start with the known facts.
2. If the premise of the implication in the clauses are known, then add conclusion to the facts.
3. Go to step 2 till
   (i) We infer the value as true or false for the query, or
   (ii) No inference can occur.

This approach takes linear time. This approach is also said to be *data-driven approach*, where we try to derive considering the available facts.

Backward chaining: In backward chaining, the processing starts with the query. So, we are moving from the goal to infer the facts that would tell us about it. If the query is true, halt. (Do nothing). But if not, then find the implications that infer the query. If all the premises of these implications can be proved to be true, we infer the query to be true. This method is also called *goal-driven method*. In turn it forms a part of goal-directed reasoning.

Comparing both the approaches, the time required in this is less than the forward one. The only reason behind this is that it goes through only the relevant facts and not others. Which method should be used ideally? It is a combination of both the methods that is used by the agent during the course of inferring.

7.6 PREDICATE LOGIC

We have already studied the propositional logic—one way that helps in knowledge representation and reasoning. In this section, predicate logic is discussed. Predicate logic also known as *first order logic* is said to be more expressive. The reason of studying other form is propositional logic becomes hard and complex, when it comes to representation of the complex environments. Later, in the chapter, a comparison is made between the two logics.

Predicate logic allows to describe the objects involved and their relationships. Consider following example:

All kids are naughty.
Suzy is a kid.
Then, Suzy is naughty.

Expressing the above example in propositional logic is difficult to have valid sentences or clauses. They can be better represented in predicate. Predicate logic is powerful tool that can express and give reasoning and it is built on the top of propositional logic, making the use of ideas.

7.6.1 Representing Facts in Logic: Syntax and Semantics

The predicate logic, as the name suggests, handles the representations in the form of predicate. Any sentence has as a subject and a predicate. For example, consider a sentence— 'The car is red'. Here, 'car' is the subject and 'is red' is the predicate. While representing, it would be $R(x)$, where R is red and x is any object. This is a very basic example. We now go into the details of representations. A sentence in predicate logic is built up from constants, predicates and functions. A sentence can be an atomic sentence or sentences connected to each other.

We can represent it in the following way:

Sentence → Atomic sentence |
Sentence connective sentence |
Quantifier variable_name,..... sentence|
~ sentence

Further, the atomic sentence comprises predicate. The predicate has terms, which could be functions, constants or variables.

Atomic sentence → Predicate (Terms)
For example, Predicate: Blue, Academic, and so on. They express the relationships.

Term → Function(Terms) |
Constant |
variable_name

Constants: Mary, A, etc. They are the objects for which we want to talk about.
Variable_name: $p, q, r, \ldots$, etc.
Function: It allows to refer to the other objects indirectly. For example, father_of

The connectives and the quantifiers are used in the formulation of the sentence.

Connectives: $\Rightarrow, \wedge, \vee, \Leftrightarrow$
Quantifiers: $\forall, \exists$

We need to make a note of the difference between the function and the predicate. As said earlier, a function is indirect reference to the objects. Predicate comprises the functions. It will be more clear in the example explained later. We start with the simple ones.

Simple Sentence

Sita is the mother of Rohan.
Representation: Mother(Sita, Rohan)

Complex Sentence

Reeta's uncle and Rohan's daddy booked a flat.

Representation: Booked (Uncle(Reeta), Daddy(Rohan))
Another example can be Father(Sita) = Ramesh

Connectives

Simple example of use of connectives can be:

Father(Rohan) ∧ Father(Shyam)
~ Dancer(Riya) ⇒ Dancer(Sita)

Quantifiers

There are two types of quantifiers—which are as follows:

1. ∀: Universal quantification (pronounced as 'for all')
2. ∃: Existential quantification (pronounced as 'there exists an x such that' or 'for some x')

Universe of discourse: There is a concept of universe of discourse in predicate logic. It is a set of all the things that we talk about. This is the set of objects that we can assign to a variable in a propositional function. For example, in a wide sense, it can be a set of integer that possibly defines the boundaries.

***Universal quantification* (∀):** Syntax: $\forall x\, P$; where P is a logical expression.

$\forall x$ is called universal quantifier and P is the scope of the quantifier. The x is said to be bound by the scope of the quantifier.

In the example discussed earlier, we had a sentence "All kids are naughty."
It can be represented as follows:

$$\forall x\ \text{Kid}(x) \Rightarrow \text{Naughty}(x)$$

This says that for every x, if x is kid, then x is naughty.

Make a note that typically, the statements containing the words 'every', 'each', 'everyone' indicate universal quantifier.

***Existential quantification* (∃):** Syntax: $\exists x\, P$; where P is a logical expression.

This indicates that P is true for at least one value. ('There exists'.)
For example, Some people are kind.
This can be written as

$$\exists x\ \text{Kind}(x)$$

This says that there exists some x that is kind.
The statements containing 'some' or 'at least one' indicate existential quantifier.
These quantifiers are treated as unary connectives and have a higher precedence over the binary connectives.

7.6.2 Instance Representation and ISA Relationships

ISA and instance play a very important role in knowledge representation. These two attributes exhibit the property of inheritance. The following sentences help us in understanding these attributes:

Shyam is an engineer.
All engineers are intelligent.

Here, intelligent and engineer are the classes. ISA shows the class inclusion. ISA (engineer, intelligent) indicates that engineer class is contained in the intelligent class. Whereas, instance, as the name suggests, indicates the membership belonging to the class. Instance(Shyam, engineer) indicates the class membership.

7.6.3 Comparison of Predicate and Propositional Logic

After studying the predicate and the propositional logic in detail, let us compare them.

In terms of representation, propositional logic requires a separate unique propositional symbol. So, for representing any particular fact, it takes many symbols. (This one must have noticed in the Wumpus World). Imagine the number of symbols it would take if there were n locations and m people and you need to represent the fact of movement of some person.

Predicate logic is rich in terms of ontology. It includes the facts, relationships and the objects, whereas propositional logic is based on facts. Predicate logic also makes a compact representation available.

With respect to the complex sentences, in the introduction part of predicate logic itself, we have shown that the propositional logic cannot handle them.

7.7 UNIFICATION AND LIFTING: INFERENCE IN FOL

Unification and lifting are concerned with the inference in first order logic (FOL). It introduces the notion of logical inference. In inferring, we need to find out the results. To achieve this, quantifiers are required to be removed. This is possible by use of propositional logic. So, the basic idea is to infer in FOL; the rules of KB are converted to propositional logic and then the propositional inference is used.

We will proceed step-wise starting from the inference rules for quantifiers to the unification algorithm.

Inference Rule for Quantifiers

While inferring from quantifiers, two common rules are used—rule of universal instantiation and rule of existential instantiation.

Rule of universal instantiation (UI): Assume that the knowledge base contains the axiom—'All students who are kind are intelligent'. This can be represented as follows:

$$\forall x \text{ Student}(x) \wedge \text{Kind}(x) \Rightarrow \text{Intelligent}(x)$$

For this example, if we want to infer for Shyam or Rohan, we would have

$$\text{Student(Shyam)} \wedge \text{Kind(Shyam)} \Rightarrow \text{Intelligent(Shyam)}$$

In the same way, it can be inferred for Rohan. What we are doing here is making the use of substitution. The rule of universal instantiation is used for substitution. The rule states that by substituting a ground term (i.e., a term without variables) for the variable, we can infer any sentence. The rule is as follows:

Let *Subs* (θ, S) denote the final outcome after applying the substitution, where θ is the substitution applied to the sentence S.

$$\frac{\forall \ v S}{Subs(\{v/t\}, S)}$$

The rule says that for any variable v of S, the substitution occurs with t, where t is ground term. It is because of this rule that we have derived the previous results. So, *Subs*{v/Shyam} was used.

Rule of existential instantiation (EI): For any sentence S, and variable v, and a constant symbol c (this should not appear anywhere in the knowledge base), the rule is given below:

$$\frac{\exists \ v S}{Subs(\{v/c\}, S)}$$

Again, take a hypothetical KB having the following sentence:

$$\exists x \text{ Car } (x) \wedge \text{Drive } (x, \text{Shyam})$$

To infer we can have Car(abc) $\wedge$ Drive(abc,Shyam).

In this sentence, by this rule, it implies that as long as the constant abc does not appear anywhere in the KB, the inference is correct. So, a new name is to be used, this is called *Skolem constant*.

One thing to mention here is that with the application of UI, the new KB is logically equivalent to the old one. UI can be applied several times, but with EI, as it is applied once, new KB is not equivalent, but satisfiable.

Reducing to Propositional Inference

In order to reduce from predicate logic to propositional, we need to have the rules for inferring the non-quantified sentences from the quantified ones. The technique of propositionalisation is discussed here. The substitution methods mentioned in the previous section are to be used. In case of existential quantified sentence, it can be substituted by one instantiation, whereas in case of universal, it is replaced with all

possible instantiations. Let us take the following KB:

∀x Student(x) ∧ Kind(x) ⇒ Intelligent(x)
Student (Shyam)
Kind (Shyam)
Friends (Shyam, Rohan)

Now, since we want to have the reduction to take place, after applying the substitution methods, we can have the representation as

Student (Shyam) ∧ Kind (Shyam) ⟹ Intelligent (Shyam)
Student (Rohan) ∧ Kind (Rohan) ⟹ Intelligent (Rohan)

The KB now has just the axioms that can be mapped as propositional symbols. The new KB would contain Student (Shyam), Kind (Shyam), Intelligent (Shyam), Student (Rohan), Kind (Rohan), Intelligent (Rohan).

Problems with propositionalisation: It is obvious to us that propositionalisation creates many irrelevant sentences. Now, we are aware of the substitution of Intelligent (Rohan). But then is there a need to generate Kind (Rohan) if in the KB it is specified Friends (Shyam, Rohan)?

This inferring generates a lot of facts. If we are having p predicates of k-ary and n is the number of constants, then the outcome of propositionalisation would be $p * n^k$.

7.7.1 Unification

It is the process of finding the substitutions that make different logical sentences look identical, i.e., the process of finding substitutions for predicate parameters. One would think that these inferences are so simple; you look at the sentences and understand. But it is required that the machine understands them, and hence, there is need of process of substitution.

For example, ∀x Student(x) ∧ Kind (x) ⇒ Intelligent (x)
Student (Shyam)
Kind (Shyam)

From this, we infer that Shyam is intelligent.

First Order Inference Rule

While inferring that Shyam is intelligent, we have applied the substitution method. First, x is found such that x is student. Then, x has to be kind and after that, we infer that x is intelligent. We have carried out the substitution of {x/Shyam}. Assume we do not have the fact of Kind (Shyam), instead we have

∀y Kind(y)

Will it be possible to conclude that Shyam is intelligent? Naturally, as Shyam is student and that everyone is kind, this inference stands true.

So, we have {x/Shyam} and {y/Shyam}.

The identicalness achieved is student(x) and Kind(x) with student(Shyam) and Kind(y). This generalised process is the rule of generalised modus ponens.

The rule states that for the atomic sentences, a_i, a'_i and q, there is a substitution θ such that $Subs(\theta, a'_i) = Subs(\theta, a_i)$ for all i.

$$\frac{a'_1, a'_2, a'_3, \ldots, a'_n, (a_1 \wedge a_2 \wedge a_3 \ldots a_n) \Rightarrow q}{Subs(\boldsymbol{\theta}, q)}$$

a_1 : Student(x)
a_2 : Kind(x)
q : Intelligent(x)
a'_1 : Student(Shyam)
a'_2 : Kind(y)
θ : {x/Shyam} and {y/Shyam}
$Subs(\theta, q)$: Intelligent(Shyam)

The generalised modus ponens is a lifted version of modus ponens. It raises the modus ponens from the propositional to FOL. This is called *lifting*.

Unification Algorithm

Now, finally coming to the discussion of unification, how do we determine the substitution of θ? The rules of inference that are lifted need to find substitutions that make different logical sentences look identical. This is called *unification process*.

The algorithm is given below:

The algorithm Unify(a, b) takes two sentences and returns a unifier for them, if there exists one. In short, we now decide the value for θ.

Unify(a,b) = θ, where $Subs(\theta,a) = Subs(\theta,b)$

Let us try to answer the query Friends(Rita,x)—who all are friends of Rita?
Assume in the KB, we have the following:

Friends(Rita,Seema)
Friends(x,Maya)
Friends(y,Neha)

Applying the unification, we get:

Unify(Friends(Rita,x),Friends(Rita,Seema)) = {x/Seema} = θ
Unify(Friends(Rita,x),Friends(x,Maya)) = Fail= θ – *
Unify(Friends(Rita,x),Friends(y,Neha)) = {x/Neha,y/Rita} = θ

The case marked as * is failed due to the use of same variable name. That is, x cannot

be Rita and Maya at same time. It is required that the names should be different. To achieve this, standardising is done. The variable name is changed, and hence, we could get

$$\text{Unify}(\text{Friends}(\text{Rita},x),\text{Friends}(z,\text{Maya})) = \{x/\text{Maya}, z/\text{Rita}\} = \theta$$

The algorithm returns substitution to make x and y look identical.
Let us write a generalised algorithm for it.

1. If x and y are variables or constants,
   (i) x and y are identical return NULL
   (ii) Else, if x is variable and x occurs in y, then fail, else return $\{y/x\}$
   (iii) Else, if y is variable and y occurs in x, then fail, else return $\{x/y\}$
   (iv) Else fail
2. If the arguments mismatch, return fail.
3. If the predicates do not match, return fail.
4. Let *Subs* = { }
5. For *ctr* =1 to the number of arguments
   (i) Go to step 1 (call again) with *i*th argument of x and y and add result to *Sol*.
   (ii) If *Sol* = fail, return fail.
   (iii) If *Sol*! =NULL.
   Again apply *Sol* to the remaining part of x and y.
   Add *Sol*: *Subs* = *Sol* + *Subs*
6. Return *Subs*

Unification plays a very important part when it comes to natural language parsers. It has a strong mathematical concept and is important in the other AI programs as well.

7.8 REPRESENTING KNOWLEDGE USING RULES

This topic essentially covers the use of rules for the representation of knowledge. After discussing the details of propositional and predicate logic, now, the rules for encoding the knowledge are discussed. We have studied about the representation of sentences in logic formats, but the rules that actually shape the knowledge are very crucial. Let us study the representations. This section essentially covers the gist of the earlier part studied in terms of reasoning, i.e., the necessities in rule-based system with KB representation.

7.8.1 Declarative and Procedural Knowledge

The need for the representation of knowledge in terms of rules is finally to get the solution to our problem. In the previous sections, we have looked into the assertions. Here, we take them further to resolve the problem.

Two types of representations exist—declarative and procedural. In case of declarative, knowledge is specified; but extent upto which the knowledge is required to be put up is not specified. The assertions mentioned in the previous section are declarative. In order to use a declarative representation, it needs to be augmented with a program that specifies what is to be done and how it is to be done. The assertions can have resolution theorem applied. These assertions can be viewed as program. While doing so, the implication statements actually define the reasoning. In simple words, we can say that the declarative representation is of facts.

In case of procedural representation, the control information required to make use of the knowledge is embedded in the knowledge. So, here, an interpreter is required that understands the instructions in the knowledge. It can have heuristic too to have the result generated. Let us take the following example of the KB:

Bird(Parrot)
Bird(Sparrow)
Feathers(Pigeon)
$\forall x$ Bird(x) $\Rightarrow$ Feathers(x)

Let us say we want to solve a query to find out some y that has feathers.
The query representation will be as follows:

$\exists y$: feathers(y)

What is the expected answer? It will give us the entire three—sparrow, parrot as well as pigeon. So, it is dependent on the order in which the assertions are actually being written. If some control knowledge is added to make it procedural, the answer that we might get is pigeon. This control knowledge could be the order in which the facts are to be evaluated, say depth first search.

Which representation should be selected? Generally, viewing the knowledge as procedural or declarative is not an essential characteristic for building the KB, but it is the method that permits for knowledge extraction.

7.8.2 Logic Programming

It is a programming language paradigm in which logical assertions are viewed as programs. It comprises facts and rules. The facts are the axioms and the rules determine whether the axioms are sufficient to determine the truth of the goal. We discuss about **pro**gramming **log**ic (Prolog) in brief to understand the paradigm. Prolog comprises Horn clauses. In Prolog, the interpreter has a fixed control strategy. The working of the control strategy is explained below:

1. It begins with the problem statement, mapping it to the final goal to be achieved.
2. Checks for assertions and rules to see if the goal can be proved.
3. For this decision to apply a rule or fact, unification process is also invoked.
4. It uses backward reasoning to get the solution.

5. It may apply depth first strategy and backtracking.
6. When it comes to the choice, consider the order while selecting the next rule or fact.

7.8.3 Forward and Backward Reasoning

In Prolog, the reasoning is backward. But in practice, there are two types of reasoning—forward and backward.

Forward Reasoning

In forward reasoning, the process starts with the starts states. A tree is built considering the different moves or intermediate steps that could be the solutions. At each point of time, the next level is generated with the selection of rule, whose left side matches with the root. The process is from left to right evaluation of a rule. This process is continued till the goal state is achieved.

Backward Reasoning

It is exactly reverse of forward reasoning, where the process starts with the goal. The rules whose right side matches with the root are considered. So, these rules define the next level that would be required for the generation. So, the process is from right to left. Once a rule is applied, its left side is then looked upon and searched in the right side again. This is continued till we reach the initial state.

Is it possible that the same rules are being used in the reasoning process? Yes. But then, which reasoning is to be applied is dependent on the topology of a problem.

In the eight-puzzle problem discussed in earlier chapters, the direction did not matter. But it does matter when

1. There are more goal/start states.
2. If new facts are likely to be evolved, then forward reasoning is better.
3. The branching factor is at the decision node.
4. Finally, the justification required may make the user to use the reasoning he/she thinks.

The two rule systems that now are taken into account are the forward and backward chaining. Prolog and MYCIN are the two systems that are under this category. A system that is directed by goals is *backward chaining*, whereas in case of *forward chaining*, it is data driven. We feel that forward chaining is very simple to process, but the rule application and matching in forward chaining are more complicated.

Is it possible to have a combination of forward and backward reasoning? The answer is yes. As an example based on some facts, if one tries to infer or conclude, especially in diagnostic cases, it is better to apply backward reasoning.

7.8.4 Matching

The question that is addressed here is how can the rules be extracted that are matched in the course of inferring a position with the current state? This needs to perform matching that is to be carried out between the current state and the preconditions of the rules. Different methods of matching are discussed below:

Indexing

A very simple thought that comes to our mind is to simply check every precondition with the current state. Are we performing linear search? Naturally, this is not the option that would be looked at because of the following two reasons—1. As discussed earlier, the large number of rules that would be existing, would end up, thereby making it the most inefficient method. 2. It is not obvious to say that whether the preconditions would be satisfied by any particular state.

The most obvious answer to this problem is the use of indexing. We need to access the appropriate rules and for that, we use index. So, an index is typically used in this case. If we have a chess board, an index is assigned to the board positions. Further, with the use of hashing technique, the key that represents the same position comprises the same rules which give a specific position. So, with the same key, all the required rules can be found. Will this method have any pitfalls? The representation has all the positions and cannot handle its generalisation. Rules can be indexed by the pieces and their positions. Though there is a drawback, indexing is a very important aspect of the matching process.

Matching with Variables

When we say that the rules are being matched, one more problem that can arise is the unmatching of the preconditions with that of the current situation. What should be done under this scenario? So, again a search is to be carried out to check whether there exists a match between the preconditions and the current state.

Generally, the unification algorithm is used to solve this, and is useful too with the single-state description scenario. Forward and backward method too can be employed here. Even unification can be applied again, but it is better to have many-many match. Here, many rules are matched against many elements. Let us discuss about an algorithm for this many-many match—RETE.

The algorithm maintains the network of the rules and it uses state descriptions to determine which rules can be applicable. This is possible, as the rules are not changed randomly and the state descriptions do not change. Structural similarity in the rules is also used. A very simple example of this is has-strips and has-spots. Actually, these predicates are structurally same. The algorithm is also persistent in case of variable binding.

The Rete algorithm is based on facts—rule pattern matching problem. It preserves information about the structure of network with matches.

The network comprises nodes that are individual condition elements. Every node has

two sets.

1. Set containing all the working memory elements, where the condition node matches.
2. Combinations of working memory elements along with the bindings that generate consistent match of the conditions.

This setup helps in avoiding repetitive testing of all rules in each cycle. Thus, the nodes impacted during addition of new facts are checked.

For example, we have rule

if $P_1(A, 5)$ and $P_2(A, C)$ then res(A, C)

if $P_1(A, 6)$ and $P_2(A, C)$ then res(A, C)

It begins with the structure, as shown in Figure 7.6.

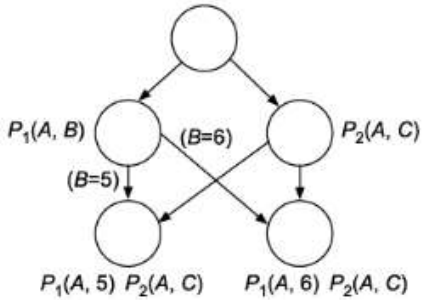

Figure 7.6 Rule pattern matching.

If a new fact $P_1(7, 5)$ is added, then it indicates mismatch in rule, as shown in Figure 7.7, where the data is propagated.

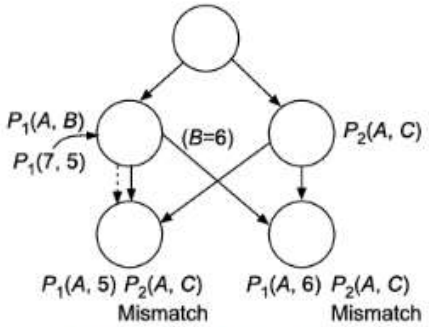

Figure 7.7 Rule propagation.

Thus, Rete algorithm performs efficient matching.

Approximate and Complex Matching

Another aspect of matching is required when the preconditions describe the properties that are not explicitly mentioned in the current state. Here, altogether a new set of rules are required to describe how only some of the properties can be inferred. There could also be a case, where we need to have the preconditions applied approximately. The most common example here is the speech. Mapping this while considering the noise and interference, it will be very difficult to approximate.

In some cases, matching the rules to the problem statement is done. ELIZA—a very old AI program that simulated the behaviour of a Rogerian therapist had used this matching. It matched the left side of rules against the last sentence a user would type. A simple dialog between ELIZA and a woman is given below:

Woman: I need your help.
ELIZA: How can I help you?

Or possibly, it would answer saying 'Tell me more about yourself', if the woman says, 'I am the only child in the family', and so on. One thing that you must have noticed is that there is a pattern matching; the patterns ask about some specific word that has been used by the woman. The entire sentence need not be matched. This approximation makes ELIZA put forth its accountability in an effective manner.

Conflict Resolution

In matching process, the rules need to be matched and it is equally necessary to decide the order in which this is to be carried out. Though it is the search method, which is responsible for matching, the decision can be built in the matching process. This stage of matching is referred to as *conflict resolution*. This preference assignment can be made on the following bases:

1. Rule that is matched
2. Objects that are matched
3. Action that the matched rule performs

7.9 SEMANTIC NETWORKS

We have already mentioned about semantic networks in the introductory section with respect to the knowledge representation. Here, we provide details of the same.

The basic idea behind semantic network is that knowledge is better understood as a set of concepts that are related. In other words, the semantic networks represent the conceptual relationships. A semantic network comprises nodes that are connected to each other by arcs. There are different variants of this network and each of them essentially does the same thing. It is noticed that the boxes or ovals are used to represent the objects or the categories of objects.

Common Semantic Relations

Though there is not a fixed set of relationships to express, we highlight the most common ones that invariably occur in every logic.

1. *Instance:* x is an instance of y, if x is a specific example of the general concept y. For example, Tendulkar is an instance of batsmen.
2. *ISA:* x ISA y if x is a subset of more general concept y. For example, batsman ISA cricketer.
3. *Haspart:* x is haspart of y, if y is part of the concept x. For example, stumps haspart bails.

Considering the same example of mapping the knowledge of some cricket domain, we can draw the semantic network. The semantic network for same is depicted in Figure 7.8.

From Figure 7.8, it is understood that the team and the runs are the values of attributes. Arrows indicate the point from object to its corresponding value. The logic for the above semantic network can be ISA(Batsman, cricketer), instance (Tendulkar, Cricketer), team (Tendulkar, India) and so on.

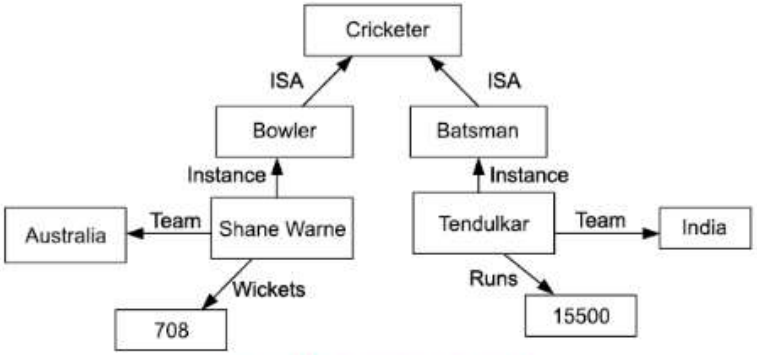

Figure 7.8 Semantic network.

ISA and instance are actually inheritable instances. Now, if we have two predicates, is it possible to map in semantic network? Definitely, we can do so. By creating the corresponding nodes and representing them with their relationships, it is possible to have a semantic network.

Inference in Semantic Network

While inferring in the semantic network, two mechanisms are followed, viz. intersection search and inheritance.

In *intersection search*, the intersection between the nodes is found, and in turn, the relationships too. This is achieved by spreading the activation and making the nodes visited. Whereas, in case of *inheritance*, the ISA and instance allow to implement this process. It provides the means for enabling default reasoning. While making the inferences, it is also required that the difference between the links (which hold values) is clearly mentioned. For example, the runs taken by players Shyam and Ram need to be explicitly differentiated, say by means of greater than link, where we need to compare them.

Figure 7.9 could be transformed into Figure 7.10. The arrow between Runs_1 and Runs_2 is the comparison of the runs.

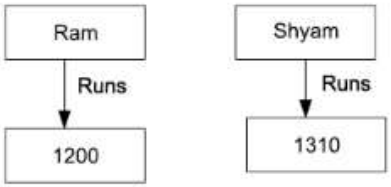

Figure 7.9 Inference with semantic network: Example.

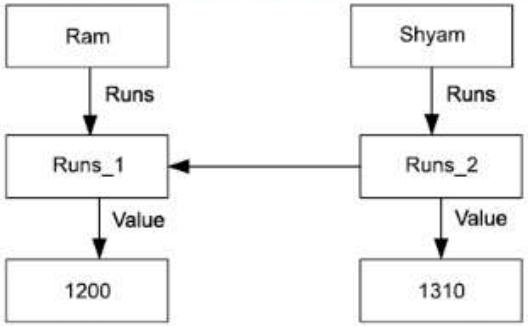

Figure 7.10 Inference with semantic network: Example.

7.9.1 Partitioned Semantic Networks

Partitioned semantic networks are the extension of semantic networks. They overcome the drawbacks of semantic networks or extend their knowledge expression. Most often, in semantic network, there exists vagueness. This occurs while finding the meaning of the tokens. The inference mechanism too is inefficient, and heuristic inadequacy exists as well. Partitioned semantic network forms a more sophisticated logical structure. Here, the network is split into spaces that consist of nodes and arcs and each space is then considered to be a node.

Consider an example, where a lawyer argues that the case is strong (Figure 7.11).

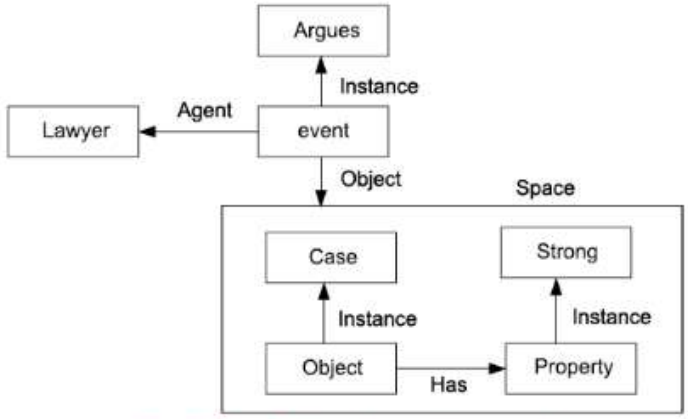

Figure 7.11 Partitioned semantic network.

From Figure 7.11, it is understood that the 'case is strong' is partitioned into a space that in turn, refers to an object or a node. So, the partitioned networks allow for the propositions, without commitment to the truth and the expressions to be quantified.

These networks enable to distinguish between the objects and to control the search space, proving them logically as well as heuristically adequate.

7.10 FRAME SYSTEMS

Frame systems or rather *frame-based systems* are knowledge representation systems that use frames. Frames are also regarded as extensions of the semantic networks. Semantic networks are concerned with labelled representations. But when the task gets more complex, what is required is a structured representation. A frame system handles this. A *frame* is a collection of attributes or slots. It is a structure to represent the concepts that have some associated value in the real world.

7.10.1 Minsky's Frame

The frame structure was proposed by Minsky in 1975. It was proposed that the knowledge should be organised into chunks, called *frames*. These frames capture the essence of the concepts by clustering relevant information. So, a large amount of procedurally expressive knowledge should be a part of the frames. Clubbing and organisation of such frames form the frame system.

Representation of Frames

Each frame has a name and slots (think of the concepts that we have studied in object-

oriented programming—OOP). A simple frame is represented below (refer to Figure 7.8):

Tendulkar
Instance: Batsman
Runs: 15500
Team: India

We can have the frames for 'cricketer'. This would be a class. The frames for 'Tendulkar' and 'Shane Warne' are instances. We can also define a meta-class, which contains the elements that are classes themselves.

7.11 CASE GRAMMAR THEORY

Case grammars provides an all together different approach to the problem with regard to the field of linguistics for studying the human languages. Thus, the theory plays a role in natural language processing. It shows that how syntax and semantic interpretations can be combined. So, it represents one more way for knowledge representation. The grammar rules are such that they describe the syntactic regularities instead of semantics, but when it comes to the structures that are been produced by them, they have semantic relations rather than the syntactic ones. We will discuss about this theory in detail in Chapter 12.

7.12 INFERENCE

Inference is a mechanism by which reasoning takes place. They go together and it is very much essential to have an appropriate inferring mechanism so as to carry out reasoning. This section, deals with the different categories of reasoning by which new conclusions are derived.

7.12.1 Deduction, Induction, Abduction

They are the basic types of inferences. We introduce the concept of each of them as follows:

Deduction

Deduction is the process of deriving a conclusion from the given axioms and facts. So, drawing a conclusion based on the available knowledge and observations is deduction. This conclusion can be drawn by applying the modus ponens rule.

A very simple example to explain this can be as follows:

Axiom: All kids are naughty.
Fact/Premise: Riya is a kid.
Conclusion: Riya is naughty.

Inference in deduction is also referred to as *logical inferring*. Deduction is often called *truth preserving* also. Calculus can be used for this inferring.

Induction

Induction is deriving a general rule (axiom) from the background knowledge and observations.

For the same example, we can say that
If we have the knowledge with us that

Riya is a kid.
Riya is naughty.

then we can have the general axiom that

Kids are naughty.

Abduction

A premise is derived from a known axiom and some observations. Consider the same example, but now the information available is different.

Information available:

All kids are naughty.
Riya is naughty.

We can infer that Riya must be kid.

Thus, we get the differences and understand the process of inferring. But which is to be used and at what point of time? Abduction is typically for the diagnostic and expert system, and one must have guessed that deduction is concerned with FOL to get the inferring.

7.13 TYPES OF REASONING

The type of decision-making and reasoning changes as per the application and domain involved. This section highlights some of the reasoning approaches.

7.13.1 Common Sense Reasoning

Common sense reasoning is concerned with the way we think. This is essential when we are representing the KB. Do the machines have common sense? We need to embed it. While doing so, we need to formalise the reasoning. This is possible with the use of formal logic. In humans, we relate common sense with the ability of taking good judgement. But in case of AI, this is the technical sense with the facts and understandings. This reasoning is required especially in natural language processing. Various tools are available for the common sense knowledge base like ConceptNet that is an extended version of WordNet.

7.13.2 Hypothetical Reasoning

In hypothetical reasoning, different assumptions are looked at to see what can be followed

or inferred from them. It can be termed as the reasoning about different worlds. When and where is the requirement of this type of reasoning? It is applied for diagnostic system, where we can assume some hypothesis and then make predictions followed by their verification. A method of hypothetical-deductive reasoning is applied in this case.

The steps involved in the hypothetic reasoning are as follows:

1. Evaluate each of the hypothesis.
2. Hypothesis is selected and tested from which predictions are generated.
3. Using experimentation, predictions are validated for correctness. If these are correct, then the hypothesis is accurate, else the hypothesis is not valid.

7.13.3 Analogical Reasoning

It is the process of reasoning from one particular object to another. Conclusions are derived from experience or similarity-based situations or conditions. For example, *P* is like *Q*. If *P* contains *A* and *B* is in *Q*, then we can infer that *A* is like *Q*. While reasoning, we find out the analogy in the given data. This is like finding the 'like' thing. In order to use the analogy

1. One starts with a target, where a new understanding is to be created.
2. Matching domain is found and further, the matching terms as well.
3. Ahead of it, related terms are found and then attributes are transferred from matching to target.

SUMMARY

The chapter highlights the need for proper knowledge representation and reasoning. Representation of the knowledge should be such that the inference is efficient. With the different ways of representing knowledge, logic administers the overall mechanism. To have a sound inference, accurate and precise logic is a must. Propositional logic and first order logic are used in the knowledge formulation. Semantic networks help in understanding the relationships and prove to be beneficial in understanding the hierarchy structure. Further, the methods of unification and lifting make the inferring simpler. The partitioned semantic networks and the frames further help in getting the knowledge representations better. Still with the known facts, it is definitely not a complicated task to handle, but with limited and uncertain knowledge, they need to be handled in a different way. The next chapter deals with the same.

KEYWORDS

1. **Knowledge:** It is the information that helps in taking decisions or solving problems at hand.
2. **Knowledge base:** Knowledge is stored in a structure called knowledge base, which is often referred to as KB. A KB is formed with the available facts and updated in the process of acquiring more knowledge that is used by an agent.
3. **Relational knowledge structure:** It is a knowledge structure where the facts can be mapped into the relations and stored in the database.
4. **Inheritable knowledge structure:** It is a structure that enables inheritance of the properties to improve the inference.
5. **Slot-filler structure:** It is a structure that belongs to the category of inheritable structure and allows easy accessibility and retrieval of the required information.
6. **Inferential knowledge structure:** It is a knowledge structure that makes use of logic in representation and maps the inferred facts.
7. **Procedural knowledge structure:** Procedures with the knowledge embedded in them form this type of structure.
8. **Knowledge based agent:** It is an agent whose actions are not arbitrary but are based on some valid knowledge. KB is used by the agent.
9. **Horn clause:** It is a clause that is disjunction of literals of which at most one positive literal exists.
10. **Unification:** It is the process of finding the substitutions that make different logical sentences look identical.
11. **Partitioned semantic network:** It is the extension of semantic networks, where the network is split into spaces that consist of nodes and arcs and each space is then considered to be a node.
12. **Frames:** It is also the extension of the semantic networks. It is a structure to represent the concepts that have some associated value with regard to the real world.
13. **Minsky's frames:** Early frame structure was proposed by Minsky. These frames capture the essence of the concepts by clustering some relevant information.
14. **Deduction:** It is the process of deriving a conclusion from the given axioms and facts.

15. **Induction:** It includes deriving a general rule (axiom) from the background knowledge and observations.
16. **Abduction:** In abduction, a premise is derived from a known axiom and some observations.

MULTIPLE CHOICE QUESTIONS

1. Given a fact and an axiom/premise, the reasoning falls under
   (a) Induction
   (b) Deduction
   (c) Abduction
   (d) None of these
2. A procedure approach that produces proof by contradiction is
   (a) Abduction
   (b) Refutation
   (c) Logic programming
   (d) None of these
3. Which of the following is involved between the mapping of initial facts to the final facts?
   (a) Forward chaining
   (b) Forward representations
   (c) Reasoning
   (d) Both (b) and (d)
4. The basic difference between the normal on simple agents and the knowledge-based agents is
   (a) Knowledge-based agents can capture the entire percept, whereas the simple cannot.
   (b) Knowledge-based agents are suited for some specific tasks.
   (c) Simple agents can take arbitrary actions, whereas knowledge base do not.
   (d) All of the above
5. For a query like 'Where is my mobile?' to be resolved, which approach is appropriate?
   (a) Forward chaining
   (b) Forward checking
   (c) Modus ponens
   (d) Backward chaining
6. Consider the following statements:
   $S(x)$: x is a primary school student.
   $I(x)$: x likes to play games on i-pad.
   Every primary student likes to play games on i-pad' can be mapped as (where universe of discourse is a set of students)
   (a) $\forall x(S(x) \rightarrow I(x))$
   (b) $\exists x(S(x) \rightarrow I(x))$
   (c) $\forall x(I(x) \rightarrow S(x))$
   (d) $\exists x(S(x) \vee I(x))$
7. Which of the following clearly defines a frame system?
   (a) An inference system
   (b) A system that maps the facts and beliefs
   (c) A form of knowledge representation
   (d) A system with a set of facts and their instances
8. Which of the following best justifies the knowledge?
   (a) It is a known information
   (b) It is a set of reasoning system
   (c) It is used for inferring
   (d) It is a set of assumptions

CONCEPT REVIEW QUESTIONS

1. What is a knowledge base?
2. Are inferring and reasoning same? Discuss.
3. What are the different representations of the knowledge base?
4. Explain and discuss semantic networks, partitioned semantic networks and frames.
5. Write the following using first order logic
   (i) Some intelligent students study intelligent systems.
   (ii) Every student who opts for intelligent systems is a determined student.
   (iii) There is at least one student of intelligent systems, who does not like the AI assignments.

CRITICAL THINKING EXERCISE

1. For the following example, write the KB in propositional logic. Explain how the inference would occur.
   Ram is an IT professional.
   IT professionals are hardworking.
   All hardworking people are wealthy.
   All wealthy people pay heavy income tax.

PROJECT WORK

1. Develop a program to determine the state of wumpus, with the initial states and the knowledge discussed in Figure 7.4. Which type of KB representation will you use and why?
2. Using semantic network, represent different players in Indian Cricket team and their properties. Write a query to determine most prolific batsman on Australian soil using this semantic network.