CHAPTER 15

Game Playing

Learning Objectives

- To understand the concepts of game playing such as game classes, game equilibrium and game playing strategies
- To understand the importance of representing games in different data structures
- To study the approaches for capturing game to be solved as search problem
- To study the searching solutions for game tree using minimax algorithm
- To acquire knowledge about state space reduction in the form of alpha-beta pruning
- To understand the application perspective and have an overview of different game theory problems

INTRODUCTION

Games have always represented a means to express power and intelligence. In computer games, it is more of the intelligence which is put to test. Game playing has developed as a science, which can be adapted to analyse various scenarios in different domains. Game playing is an important aspect of a system, where two or more entities compete to achieve a state of maximum gain at the cost of other. We can identify players or entities struggling to maximise their own gains and inflict loss to the opponent. A system can be thought of as an environment, which decides the rules and permissible states. In the environment, there can be teams or an individual. A player or a team is bound by some set of rules that need to be followed for the game. Permissible moves within the framework of rules may result in gain or loss to the players. Violation of rules can be flagged as non-permissible moves and may result in negative reward as punishment.

For example, think of an ecosystem in equilibrium, where each living being struggles to survive. Nature has decided the environment and ecosystem. Depending upon the various factors, even the environment may change. We can visualise the changing environment as permissible moves by nature, and hence, nature can also be seen as a player in the ecosystem. In the ecosystem, a lion with strong muscles and jaws chases a deer, which reflects its plausible moves. A deer is equipped with highly sensitive sensors and is capable of maneuvering through the bushes to escape attack of lion. Too many wins for lion will result in the extinction of deer and amount to depletion of food stock for lion, and ultimately, it will also die out. Too many wins for deer will result in starvation of lion, and hence, it may perish. No rain will result in drought and force deer to abandon or perish in the ecosystem. This will in turn result in starvation among lions and they will tend to migrate towards greener pastures. The game between nature and players of the ecosystem goes on. Nature tries to create a balance between the players of the system by controlling the environment of ecosystem. It has also provided the power of adaptation to living beings for survival in the modified environment. The fittest survives by developing new skills. The weaker perishes for not being able to adapt. The game of struggle and survival is just an example, where multiple players survive by enhancing the chance of survival at the cost of others.

The essence of opportunistic gain appears in various fields around us. Think of a game played by police and thief in the environment called society. Society frames the rules. Police enforces the rule and a thief tries to maximise his/her gain by breaking societal norms. People from different cultures try to enforce their ideology on others; it is again an example of game playing for survival or to prove oneself as the best. Politicians try to maximise their influence base on the basis of social structure, which is again at the cost of opponent by social reengineering. There are various fields such as economy, biology, computer science, interrelations, human behaviour etc., where we find several entities struggling to enhance their gains.

Over a period of time, game theory has developed as an advance tool in mathematics and has been applied to various fields in order to have foresight before taking any decision, under a given scenario, which may be captured in the state of the game. Even before a game is played, various states, permissible moves, counter moves of opponents, gain and loss related to moves, etc. can be visualised, and hence, the best possible move can be made so as to force the opponent into a state of disadvantage. The player's behaviour in strategic problem can be modelled in proven mathematical terms, in which success of a player A depends on the move of player B. Formalisation of game theory began in 1944 through a book 'Theory of Games and Economic Behaviour', authored by

Return to page 332

John von Neumann and Oskar Morgenstern. Since then, the art of game playing has been adapted to several fields and has developed as an advanced tool in artificial intelligence because of its capability of foresight in a decision-making process carried out by a player.

Game theory has helped the economists to get insight into behaviour patterns of various organisations and their customer interactions and have an analysis of the transactional information. The focus of this analysis is on the maximisation of the utilities. Other areas where it is put to work include political sciences, other behaviour and social aspects of humans and biology too. When we say about political science, it is more about the application of the theory in fair division, political economy, public choice, war bargaining, positive political theory, and social choice theory. In regard to the biological domain, the concept of payoffs is mapped with the fitness. The main focus has been on evolutionary aspects of life. Additionally, biologists have used evolutionary strategies for the game theory that would highlight the existence of animal communication. What is the use and impact of this theory with regard to machines? Talking about logic, the theory is essential for prediction purposes. The *predictions* are the moves that possibly the opponent is likely to execute. Besides, the researchers in the domain of computers have applied this theory to the model interactive computations. Game theory provides a basis for modelling and behaviour analysis of multi-agent systems. Approaches have been developed for equilibrium in games, markets, computational auctions, peer-to-peer systems, security, etc.

15.1 IMPORTANT CONCEPTS OF GAME THEORY

A game may have several facets. Some of the important concepts related to games are set of rules, number of players, plausible strategies for players, state of the game, equilibrium, information about moves of opponent, turn to make move, probabilistic approach, finiteness of turns, etc., which are to be considered while describing a game. In this section, we discuss some of the important concepts related to the game theory such as game classes, game strategies and game equilibriums.

15.1.1 Game Classes

Diversified approach has resulted in several classes of games. Some of the important classes are as follows:

1. *Symmetric game:* In symmetric game, the gains for playing a specific strategy is dependent on the other strategies or the techniques that are in practice. It is independent of the players, thus it is possible to differ the identities without impacting the gains.

2. *Zero-sum game:* In zero-sum game, participant's gain or loss is balanced by losses or gains of other participant(s). If the total gains of all the participants add up to G, and the total losses incurred by all is L, then, $G - L = 0$. Zero-sum can be thought of as a constant sum, where the benefits and losses to all players sum to the same value. It can be thought of as a closed/constant system game with no external source or sink of resources. Contrary to this, in *non-zero-sum game*, the interacting players aggregate gains and losses, which is either less than or more than zero. The non-zero-sum game provides situations in which participants may lose or gain simultaneously. Other situations in the non-zero-sum game arise when the sum of gains G and losses L, of the players are sometimes more or less than what they began with. Thus, in non-zero-sum game we have two possibilities— $G - L < 0$ or $G - L > 0$, where, G is cumulative gain of all the participants and L is cumulative loss of all the participants.

3. *Perfect information game:* A game belongs to perfect information category, if all moves taken till any point of the game by all players are known to all players. Chess, tic-tac-toe and go are the examples of games with perfect information. On the other hand, several card games represent games with imperfect information.

4. *Simultaneous game:* In simultaneous game, the action/move selected by the player is independent of the move of the other players. Thus, the player is unaware and has no information about the other player's moves. A matrix, which is often called *Normal form* is used for the representation of the possible moves and the outcomes. Prisoners' dilemma is a classic example of simultaneous game.

5. *Sequential game:* In sequential game, one player takes the move before other players. The rest of the players get some information of the first player's choice. Extensive form representation can be used for sequential games. Most of the combinatorial games are sequential in nature and are often solved by backward induction.

6. *Repeated game:* This type of game has repetitions of some base game known as *stage game*. Stage game is a well-known two-player game. A player considers the effect of his current move on the future moves of the other players. Since a player plays the game repeatedly with the same player, possibility of retaliation exists, and hence, equilibrium strategy differs from the other popular games.

7. *Signalling game:* There are two players in signalling games—a *transmitter* or often called the *sender* and the *receiver*. In this type of game, incomplete information exists. Often one player is more informed than the other. Let us say that the sender S has to take decisions regarding some sort of signalling. Signalling is nothing, but an action. The receiver is less informed and it has to take decisions based on this, as this signalling could be strategically decided by the sender. The two players earn rewards depending upon the sender's signalling. A very simple example for this type could be the case of used cars, where a dealer could take decision whether some sort of warranty should be given or not.

8. *Large poisson game:* It is a game with a random number of players N, which is a Poisson random variable. The type t_i of each player is selected randomly, independently of other players' types from a given set T. Each player selects an action and then the gains are determined. Voting procedures can be modelled by such games.

9. *Non-transitive game:* It is possible that non-transitivity can arise, where multiple strategies are involved. Non-transitive game arises when the multiple strategies generate loops. For example, if in a non-transitive game, preference for some strategy A over strategy B is given and preference for strategy B over strategy C is also given, then it does not necessarily state that strategy A is preferred over strategy C.

10. *Global game:* This game too falls in the category of games in which complete information is not available. It is based on some sort of probabilities and signalling about the environment. Global games find their applications in the analysis of financial market crisis.

15.1.2 Game Strategies

Move is an action taken by a player at some point during the progress of a game. *Strategy* is a complete approach for playing the game. Generally, the strategy adopted by a player is the entire plan for the move to be executed for the situations. Strategy fully determines the player's behaviour. Some of the important strategies of the game are as follows:

1. *Dominant strategies:* Strategic dominance occurs when one strategy is better than another strategy for one player, irrespective of opponent's game.

2. *Pure strategy:* This strategy provides a complete approach for a player's game plan. It determines the move a player will make for any situation at any point of the game.

3. *Mixed strategy:* In mixed strategy, probability is assigned to each pure strategy. This allows for a player to randomly choose pure strategy from a set for a given situation.

4. *Tit for tat:* Initially, the player is co-operative. If the opponent betrays, then in the next cycle, the player chooses to betray. It is an effective strategy in iterated prisoner's dilemma. A player using this strategy initially co-operates and then responds in kind to the opponent's previous action. If the opponent has been co-operative previously, then the agent is co-operative. If not, the agent is also not co-operative.

5. *Collusion:* It is illegal and back door agreement between multiple parties (more/equal to two) to impose restriction with regard to the use of rights. It leads to uncertainty about behaviour of colluding entities in the mind of non-colluding entities and may throw surprises for non-colluding entities during the game, for which it may not be prepared.

6. *Backward induction:* It is a process that constitutes reasoning in a backward direction to identify optimal actions to be executed in the problem.

15.1.3 Game Equilibrium

Game theory attempts to find the state of equilibrium. In equilibrium, each player of the game adopts a strategy, which he is unlikely to change. Equilibrium depends on the field of application, although it may coincide. Several game equilibriums have been proposed to represent the scenario. Some of the important equilibrium concepts mentioned in literature are Nash equilibrium, Sub-game Perfection, Bayesian Nash, Perfect Bayesian, trembling hand, proper equilibrium, correlated equilibrium, sequential equilibrium, Pareto efficiency, self-confirming equilibrium, etc. Here, we discuss the most popular concept of Nash equilibrium.

Nash Equilibrium

Equilibrium is about selection of a stable state. In case of Nash equilibrium, it involves multiple players (two or more) and the following rules:

1. The participants (players) in the game are aware of the equilibrium strategies of the other participants.
2. None of the participant can gain by selecting or changing the strategy unilaterally (that would affect only one player).
3. If a participant has taken up or selected a specific strategy, then none of the other participants can get any gain by means of change in their strategy (where the other participants do not have any change in their strategy).

The situation discussed above with a set of strategy choices and their gains (payoffs) forms the Nash equilibrium.

Let us consider an example that there are two players involved in the game, say X and Y. In Nash equilibrium, X selects a strategy or a decision that is best for him considering what decision Y would be taking. Similarly, Y looks from his perspective to go for the best decision in his terms considering what X's decisions would be.

Let us take another example. Rohan and Mohan are mischievous students. They are caught by the school authorities for doing some pranks. Each of them is brought in front of the Principal separately. The Principal states that they are suspended for 2 days. This is notified to them individually. But then, the Principal feels that they are involved together in another mischief also, where a student's cycle was punctured. He speaks with them individually and tells them that if one of them confesses and the other does not, then he would suspend the one who has confessed for just 1 day, whereas the other one for 5 days. He also mentions that if both of them confess, each would be suspended for 3 days, whereas if both deny, both would be suspended for two days surely. Now, irrespective to what other's decision is, let us look from Rohan's perspective.

Return to page 332

1. If confess, when Mohan has also confessed: 3 days of suspension
2. If confess, when Mohan has denied: 1 day of suspension
3. If deny, when Mohan has confessed: 5 days of suspension
4. If deny, when Mohan has denied: 2 days of suspension

He would like to be safe. So, he would go for confess (1 and 2), where in the worst case, he would get 3 days of suspension. The same perspective exists for Mohan. The same can be represented in the form of payoff matrix (normal form), as shown in Figure 15.1 (This matrix is detailed in the further sections).

		Mohan: Confess	Mohan: Deny
Rohan	Confess	3/3	1/5
	Deny	5/1	2/2

Figure 15.1 Payoff matrix.

Thus, the confession forms Nash equilibrium. (This is the example of prisoner's dilemma that we would be studying in detail later). But do make a note that in Nash equilibrium it does not necessarily mean that the best cumulative payoff or gain is achieved for the players. Hence, it is a set of strategies in which none of the player can simply perform well by unilaterally changing the strategy. With prior knowledge of other strategies, where the others do not change the strategies but one player is able to get benefit by changing his strategy, then it is not Nash equilibrium. So, the set of strategies where the player decides not to go with the other option or perform a switch come under the Nash equilibrium. Hence, the strategy selected comes out to be the best possible response to all other strategies.

There is a concept of *Pareto optimal solutions*. These solutions are the ones in which it is not possible to make a participant, better off without dooming the other participant. An outcome is Pareto optimal, if there is no other outcome that all participants would prefer. An outcome is Pareto dominated by another outcome, say P, if all players would prefer the other outcome P.

15.2 GAME PLAYING AND KNOWLEDGE STRUCTURE

Capturing any computational problem for efficient processing has always been a challenge across various domains. The field of game playing is not an exception. Structural knowledge is basic to problem solving. It is required for creation of plans and strategies, setting conditions for different procedures, and for determining what to do when failure occurs or when a piece of information is missing. The field of game playing involves high level of planning and needs specific data structures to handle various categories of games. In this section, some of the knowledge structures are covered, with specific emphasis on game theory. Depending on the category of the game, a particular knowledge structure may suit better for a given problem. Now we are aware that any game (a problem is mapped to be a game) comprises participants or a set of players. Along with them, there are different strategies that they would be selecting and the payoffs or the gain they would achieve.

Knowledge structure is capable of capturing different aspects of the game. Some of the important structures depending upon the form of games are discussed in the subsequent sections.

15.2.1 Extensive Form Game

This form of game is used to acquire games in an order and to have the representation in the form of a game tree. The nodes in the tree inform about a decision or a choice from the player's perspective. The branches of the tree specify the strategy or the move for the player, whereas the leaf nodes represent the payoffs. A simple game tree representation is shown in Figure 15.2.

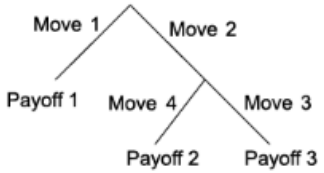


Figure 15.2 A game tree.

Let us assume that there are two players A and B. Player A is to move first. At level 0, we have the root of the game tree and first player has an option of selecting the strategy. In this case, at level 0, player A has strategy set as [Move 1 or Move 2]. Similarly, at level 1, it is B's turn to select a strategy. The leaves can be given payoff weights P_m = [Gain of A: Gain of B] for the m leaf node. It represents gain of respective players in case the game ends at the node m.

15.2.2 Strategic Form Game

As we have observed in the example of Nash equilibrium, the normal form representation is shown with the help of example of Payoff matrix. This depicts the strategies and the

Return to page 332

associated gains or the payoffs. The same can be represented in the form of some function that is able to acquire the payoffs and the corresponding strategies. Player A and B may have same or different set of strategies. The moves may be simultaneous, with each player not aware about the move of the other player.

In Figure 15.3, there are two players A and B; one player is represented across rows (say A) and other down the column (say B). Number of rows for player A will be same as the number of strategies for A and the number of columns for player B will be same as the number of strategies for player B. The payoffs are marked as pairs (G_A: Gain of A, G_B: Gain of B). For instance, if player A makes a move 'red' and player B makes a move 'black', then A scores 10 and B scores 20. On the contrary, if B makes a move 'white', then 'A' scores −10 and B scores −20. Often when the game is represented in the normal form, it is assumed that the action selected by any player is independent of the action of the other players. In the sense, the player is unaware about the next action of the other player. Whereas, if the players have knowledge about what choices others are likely to opt, the presentation of the game is in extensive form. It involves sequencing of the actions, all payoffs or gains, information available and so on. More than two players can be captured through multiple dimension representation by extending the above structures.

	Player B chooses black	Player B chooses white
Player A chooses red	G_A, G_B = [10, 20]	–10, –20
Player A chooses green	20, 10	0, 0

Figure 15.3 Payoff matrix.

15.2.3 Co-operative/Coalition Form Game

In a co-operative game, coalitions consisting of players may behave in co-operative manner to achieve the target. Hence, the game is all about having a competition between the coalitions of participants or the players. Coalition partners may agree on any aspect of the game without any restrictions. There is a transferable utility (TU), which allows side payments to be made. TU is passed to a coalition which may be distributed among players as per their agreements, which may differ from coalition to coalition. Side payments may be used as an encouragement for some players to use certain mutually beneficial strategies. There is a characteristic function to determine payoffs to each coalition. In a coalition game with $n \geq 2$ players, labelled from 1 to n, a set of players may be denoted as $P = \{1, 2, 3, ..., n\}$. A coalition C is a subset of P, $C \subset P$ and the set of all coalitions will be 2^P. The set P may be visualised as a grand coalition. Even a null coalition may be allowed for the sake of completeness. For two players, $n = 2$, four coalition, [null, {1}, {2}, {1,2}] are possible. For three players, there are 8 coalitions possible, namely, [null, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}]. Similarly, for n players, the set of coalitions, 2^P will have 2^n elements. As can be seen from the formation of coalitions, there can be members who may be shared among coalitions. However, ideally, disjoint coalition may be expected.

A coalition for game with n person can be defined as a pair (P, g), where $P = \{1, 2, 3, ..., n\}$ is a set of players and g is a real-valued function called *characteristic function* reflecting power of coalition of the game defined for all the coalitions such that

1. $g(\text{null}) = 0$
2. If A and B are disjoint coalitions ($A \cap B = \varphi$), A and B both belonging to 2^P, then $g(A) + g(B) \leq g(A \cup B)$. This is termed as *super additive property of coalitions*.

The function $g(C)$ reflects power of a coalition $C \in 2^P$.

15.3 GAME AS A SEARCH PROBLEM

Any intelligent game results into several possible moves from all the involved players. At any stage of the game, several options are available to a player for making the next move. Optimally correct choice in a state may result in superior gains. For this, a foresight is needed before initiating a move. In game theory, this foresight can be achieved by means of generating search trees, and comparing profit/loss function. The deeper the tree, the more is the foresight and more refined is the result. Unfortunately, a game tree may result into infinite search tree. Therefore, search algorithms have to limit the depth of the tree, and a comprehensive search approach may not be feasible in a reasonable time for the games resulting in infinite moves. Several AI search techniques such as iterative deepening, A*, DFS, secondary search, etc. have been successfully applied to different categories of games. In this section, we discuss the most popular game search technique minimax and subsequently discuss its refinement in the form of alpha-beta approach.

In order to formalise and understand the game theory, let us consider any board game with two players. Various components of the game would be the board itself. The initial state of the board reflects the starting configuration of the board and the players ready to move. It also includes a successor function which provides a list of legal moves corresponding to a state (move, state), a terminal test which finds out if the game has ended in a particular end state, and a payoff function which gives the value to the terminal gains in a state. The game can be represented as a search problem using these components.

15.3.1 Minimax Approach

Minimax is a decision rule that is used in the game theory so as to reduce the possible loss while maximising the potential gain. It can also be thought of as maximising the minimum gain (maximin). The rule was originally developed for two-player zero-sum game theory for two cases—players make simultaneous moves and players make alternate moves. Each player minimises the maximum gain possible for the other player. Being zero-sum game, a player also maximises his own minimum gain. However, minimax and maximin are not equivalent. Maximin may be used in non-zero-sum game scenarios.

Minimax Theorem

It was given by John von Neumann. For every two-person, zero-sum game with finite strategies, there exist a value V and a mixed strategy for each player such that

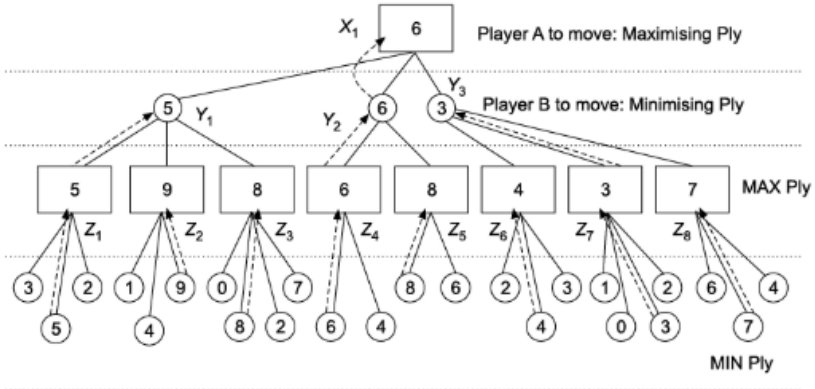
1. Given player B's strategy, the best payoff possible for player A is G.
2. Given player A's strategy, the best payoff possible for player B is $-G$.

That is, player A's move assures him gain of G, irrespective of player B's move, and similarly, player B can assure himself a payoff of $-G$.

In two-player combinatorial game theory (CGT), players make alternate moves in a defined way or strategy to achieve a defined winning condition. Game position is known to both the players, and the set of available strategies is also public. Such approach can be applied to the games like chess.

The minimax game tree can be thought of as a game tree with alternatively maximising ply and minimising ply. The root is generally taken as maximising ply with respect to the first player making a move. At a MAX level, the player tries to make a move which will maximise its gain. At the MIN level, the player tries to pass on the minimum return to the opponent.

In Figure 15.4, children of node at level Z are already evaluated. Our aim is to evaluate the maximum gain that can be achieved by player A at node X_1. Node X_1 is a maximising node. Nodes Y_1, Y_2 and Y_3 at Y level are minimising nodes. Being at maximising level, node Z_1 is assigned a value of 5 and $Z_2 = 9$, $Z_3 = 8$. Trying to return a minimum gain to Player A, at Y level, Player B can select a move limiting the gain of Player A to 5. Hence, Y_1 evaluates to 5. Similarly, nodes $Y_2 = 6$ and $Y_3 = 3$. Since X_1 is a maximising node, it has an option of selecting option, which will lead it to gain of 6. By increasing the depth of the game tree, more insight can be obtained; however, for infinitely large trees, this is a difficult task because of the complexity of number of plausible moves.



Player A and B to move alternatively,
Minimax tree evaluating value for node X_1 with respect to maximising player A

Figure 15.4 Minimax algorithm node evaluation.

15.3.2 Minimax Algorithm

The regular algorithm of minimax (as presented in the above discussion) requires dealing with minimising and maximising players separately. However, an important property, i.e.,

$$\text{MAX}(x, y) = -\text{MIN}(-x, -y)$$

can help in achieving the same result with one common evaluation function, thereby reducing the programming effort. The approach results in negmax version of minimax algorithm, which is presented as below:

```
int minimax(node, depth) // negmax version of minimax algorithm
Step 1: if (leaf node) or (depth<=0) return(node value);
Step 2: Initiate alpha = – ∞;
Step 3: for child in node{
    alpha = MAX(alpha, –minimax(child, depth–1))
    }
Step 4: return (alpha);
```

In Figure 15.5, a game tree is given with leaf nodes evaluated. At node A_1, a maximising player wants to evaluate its optimal gain.

Return to page 332

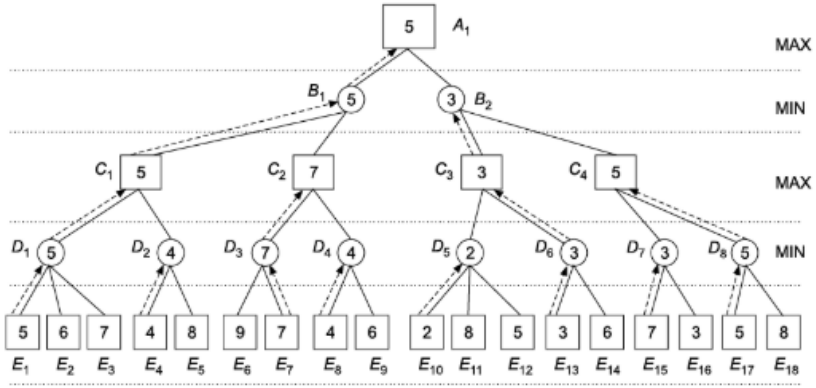


Figure 15.5 Minimax algorithm execution.

EXAMPLE: Trace the game tree to find out the optimal value for the node A_1, using minimax algorithm as given above.

Solution: Partial trace for the game tree is given in Figure 15.6. Readers are advised to traverse the complete tree for evaluating intermediate values also.

AT D_1: Minimising Node, depth = 3	
Step 1	Leaf node = false, depth==0 is false
Step 2	alpha= –15
Step 3	For(child=[E_1, E_2, E_3]){ alpha=MAX(alpha= –15, val(E_1) = –minimax(null,4–1) = –5, val(E_2)= –minimax(null,3) = –6, val(E_3)= –minimax(null,3) = –7)};hence alpha = –5
At Node D_1, we get D_1 = –5	
Similarly, at Node D_2, we get D_2 = –4	
At C_1: Maximising Node, depth = 2	
Step 1	Leaf node = false, depth==0 is false
Step 2	alpha= –15
Step 3	For(child=[D_1, D_2])(alpha=max(alpha=–15, D_1 = –(–5) = +5, D_2 = –(–4) = +4)); hence, alpha = 5
At node C_1, we get C_1 = 5	
Similarly, at node C_2, we get C_2 = 7	
Similarly, at node B_1, we get B_1 = –5	
Similarly, at node B_2, we get B_2 = –3	
Similarly, at node A_2, we get A_2 = 5	

Figure 15.6 Trace for minimax algorithm with negmax property.

15.4 ALPHA-BETA PRUNING

Minimax approach discussed in earlier section amounts to exhaustive search of solution space. In a realistic game of even modest complexity, the search space can be extremely large. It is possible to heuristically reduce the search space for games having minimax solutions. Alpha-beta pruning algorithm provides mechanism to decrease the number of count of the nodes (thus impacting the search space) in minimax tree. Most common examples of games where this is used are chess and tic-tac-toe. Further, search is stopped once it encounters a branch where at least a single value has been found that confirms that the branch would be worst in comparison to the earlier payoffs. Thus, these actions or moves should be avoided, and hence, can be pruned. Alpha-beta pruning is a perfect optimising technique and is assured to return the same result in spite of pruning. Strength of alpha-beta pruning lies in the fact that it is possible to discard the branches of the search tree without impacting any of the decisions. This results in reduction of the search time and further, it is possible to have a deeper search allowing a greater depth of sub-tree to be scanned. The algorithm comprises two values—alpha and beta. Alpha value holds the minimum (lowest) score that the maximising player is assured of. Beta value holds the maximum score that the minimising player is guaranteed to earn. The initial condition is equal to $-\infty$ and $= +\infty$. As the search progresses, this difference becomes smaller. When beta (β) value becomes lower than alpha (α), it signifies that the search beyond the current node can be restricted.

Algorithm for Alpha-Beta Pruning

The algorithm given here is a recursive algorithm for alpha-beta pruning. Parameters passed to the algorithm are as follows:

1. Player (maximising or minimising player)
2. Position (the node being evaluated)
3. Alpha (value of alpha at the node)
4. Beta (value of beta at the node)

Depending on the nature of player, algorithm returns alpha or beta value (under cut- off situation or otherwise).

```
alpha_beta(player,position,alpha,beta){
Step I: if(game decided in current board position) return (Winning Player)
Step II: children = all permissible moves for a player from node
Step III: if(Maximizing Player to Play){
      for (each child){
```


Return to page 332

```
        Step III.1:   value = alpha_beta(other player, child, alpha, beta)
        Step III.2    if (value > alpha) alpha = value; // update alpha
        Step III.3    if(alpha >= beta) return (alpha); // ALPHA CUT-OFF FOUND
        }
        Step III.4: return (alpha); //Best move in alpha
        }
Step IV:    else (Minimizing Player to Play){
        for (each child){
        Step IV.1:    value = alpha_beta(other player, child, alpha,beta)
        Step IV.2:    if (value < beta) beta = value; // update beta
        Step IV.3:    if(alpha >= beta) return beta; // BETA CUT-OFF FOUND
        }
        Step IV.4: return (beta); //Opponent's best move in alpha
}
```

Let us consider the game tree, as shown in Figure 15.7. Leaf nodes at level E reflect return values from the respective nodes and are considered to be solved. Shaded nodes are pruned using alpha-beta pruning as presented above. We trace the above algorithm to demonstrate working to achieve α-cut and β-cut.

At minimising Level D, in DFS manner, consider the node D_1. The node has received [α = –15 and β = +15], as infinity signifying the value of gain and/or loss is within the limit. From Step II, we get [E_1 = 5, E_2 = 6, E_3 = 7]. Since D_1 is a minimising level node, we use sequence in Step IV to evaluate it further.

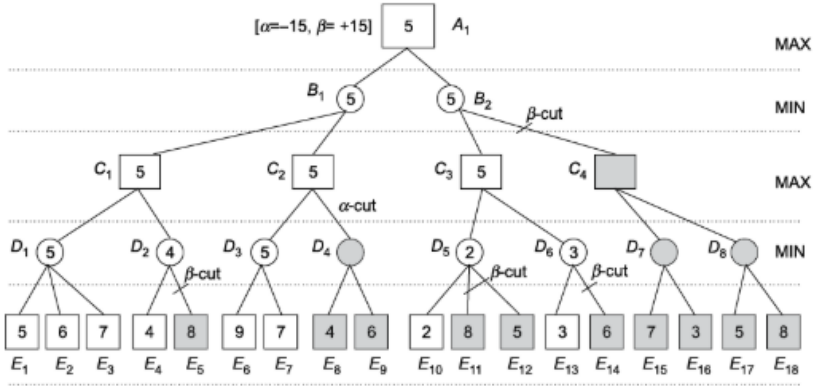


Figure 15.7 Alpha-beta pruning.

[α = –15; β = +15]

From Step IV.1: value = 5; //as returned by E_1.

Step IV.2: (value < beta) is TRUE, hence beta = 5;

[α = –15; β = +5]

Step IV.3: (alpha >= beta) is FALSE, hence child E_2 to be picked and Step IV continues.Step IV continues with node E_2 as next child of D_1.

Step IV.1: value = 6;

Step IV.2: (value < beta) is FALSE;

Step IV.3: (alpha >= beta) is FALSE, hence child E_3 to be picked and Step IV continues. Step IV continues with node E_3 as next child of D_1.

Step IV.1: value = 7;

Step IV.2: (value < beta) is FALSE;

Step IV.3: (alpha >= beta) is FALSE and next child is empty, hence move to Step IV.4

Step IV.4: return(beta=5)

Note IV.4: The value returned in the above step is the evaluated value at the node D_1 which has picked the minimum value returned by any of its opponent's move in the form of children E_1, E_2 and E_3. Same would have been done by minimax algorithm also as of now.

The trace moves to node C_1 (Figure 15.8) which is a maximising node with children [D_1 = 5, D_2 = ?]. Recursion has unwounded and hence, [α = –15; β = +15].

Return to page 332

At C_1: Maximising Node			
[α = –15; β = +15]	Step III.1	value = 5 // from D_1	
	Step III.2	value > alpha is TRUE	alpha= 5
[α = 5; β = +15]	Step III.3	alpha >= beta is FALSE	Continue
At next child D_2: Minimising Node			
[α = 5; β = +15]	Step IV.1	value = 4 // from E_4	
	Step IV.2	value < beta is TRUE	beta = 4
[α = 5; β = 4]	Step IV.3	alpha >= beta is TRUE	beta-cut has been found
	Step IV.3	return(beta = 4) as value of D_2	returns to node C_1
At C_1: Maximising Node with D_2 = 4 and [α = 5; β = +15]			
[α = 5; β = +15]	Step III.1	value = 4 //from D_2	
	Step III.2	value > alpha is FALSE	
	Step III.3	alpha >= beta is FALSE	next child is null
	Step III.4	return(alpha = 5)	Control is transferred to B_1 with C_1 = 5, and [α = –15; β = +15]

Figure 15.8 Trace for alpha-beta pruning.

The trace continues resulting in overall four beta-cuts and one alpha-cut. Readers are suggested to trace the remaining cuts and have a feel of working of the algorithm.

15.5 GAME THEORY PROBLEMS

In the chapter we have gone through various concepts related to gaming theory that have been put to application to create state of the art gaming software in different challenging fields. Most popular gaming programs are chess, backgammon, go, checkers, bridge, poker, tic-tac-toe, and others. Several intelligent and logical scenarios arising in real life have also been addressed using game theory. Some of the important problems are prisoner's dilemma, coordination game, chicken, centipede game, volunteer's dilemma, dollar auction, battle of the sexes, pirate game, dictator game, deadlock, diner's dilemma and Nash bargaining game. Computer programs have been designed to simulate the scenario in effective ways.

15.5.1 Prisoner's Dilemma

This is analogous to what we studied in the Nash equilibrium. In this section, we discuss prisoner's dilemma, which is capable of capturing several concepts related to game theory and thought process. It has a flavour of logical thinking. The problem of classic prisoner's dilemma (PD) is stated as follows:

Scenario

Two suspects, for example, A and B have been arrested by the police. The police is short of any conclusive evidence against them. Hence, it decides to separate both the prisoners. Individually, they are offered same deal by the police. The deal has the following points:

1. Here is a case where the suspects take up different decisions (or options). If one of the suspect, say A defects from the other suspect B where the other (B) does not speak up anything, the one who betrays (A) will be set free whereas the other (B) would get 14-year life imprisonment.
2. It is a behaviour where none of them speak up. Here, if A and B both remain silent, their sentence tenure comes to just 6 months of imprisonment. This is due to lack of evidence.
3. If each of them betrays the other (no common conclusion), each receives a 7-year sentence.

The suspects have following conditions:

1. Each of them has to select one of the two options—to betray the other or not to speak up (to remain silent).
2. Each of them is given assurance that the other suspect will not be provided with the information about the betrayal.

Since they are now prisoners and they are under the dilemma of which option to select, it is known as the *prisoners' dilemma*. So, they have to decide their action.

Since there is no communication between the prisoners (simultaneous game), in its standard form, each player feels safe to defect, thereby tries to maximise his own gain. In the classical scenario, the only possible equilibrium for the game is for all players to defect. No matter what the other player does, one player always gains a greater payoff by playing defect. Since in any situation, playing defect is more beneficial than co-operating, all rational players play defect.

If each player tries only to minimise his imprisonment, then the prisoner's dilemma is said to be a non-zero-sum game, where the players involved may co-operate with each other or defect from (betray) the other player. Although rational choice leads the two players to defect each other, however, the individual reward gets maximised if each player co-operates.

There are various variants for the prisoner's dilemma capturing various facets of the gaming theory. Iterated prisoner's dilemma can be played repeatedly with the same set of prisoners. Hence, the player gets a chance to punish the other player for the prior play that would possibly be non-co-operative. This format may have finite or infinite cycles. Applications of problems can be extended to economics,

Return to page 332

political science, law and study of human behaviour.

The prisoner's dilemma is shown below in Figure 15.9:

Prisoner B / Prisoner A	Co-operates with A (Silent)	Defects against A (Betrays)
Co-operates with B (silent)	Each serves 6 months (imprisonment)	Prisoner A: 14 years (imprisonment) Prisoner B: Go free
Defects against B (betrays)	Prisoner A: Go free Prisoner B: 14 years (imprisonment)	Each serves 7 years (imprisonment)

Figure 15.9 Prisoner's dilemma problem.

Irrespective of what the other prisoner chooses, each player always receives a higher payoff (lower sentence) by playing betrayal. Prisoner A is sure about "No matter how prisoner B plays, I will be better by betraying rather than remaining silent". Therefore, to maximise his own reward, player A betrays. However, if the other player acts in the similar fashion, then they both defect and both end up getting 7 years imprisonment, which is lower payoff than they would get by co-operating. Rational self-motivated decisions result in each prisoner being worse than selecting an option that would reduce the other suspect's sentence, though his own sentence is increased. Ironically, if both players co-operate with each other, their term of punishment will be only 6 months, which could be optimal. However, both would tend to prefer betrayal and end up with 7 years of imprisonment. This reflects Pareto domination of betrayal by both the players.

15.5.2 Rock-Paper-Scissors

Rock-paper-scissors is a two-player zero-sum game. Let the two players be A and B. At their turn, players produce, paper, a pair of scissors, or a rock by a gesture of the hand. Rock beats scissors, since it can crush it. Rock can be covered by paper, and hence, it is beaten by paper. Scissor beats paper, as it can cut it. The round is a draw if both A and B choose the same item.

The game can be represented by payoff matrix as given in Figure 15.10.

	Player B			
	Moves	*p*	*s*	*r*
Player A	*p*	0, 0	–1, 1	1, –1
	s	1, –1	0, 0	–1, 1
	r	–1, 1	1, –1	0, 0

Figure 15.10 Payoff matrix for rock-paper-scissors.

From the payoff matrix, it is evident that if the game is played in sequential (Rock-paper-scissors is a simultaneous game, but it is not played simultaneously) form with second player knowing the choice of the first, it is the first player who will always loose. If player A chooses Paper *p*, B will respond with scissors *s*, inflicting a loss of −1 to player A. Similarly, on choice of scissors *s* by player A, player B will choose rock *r*, to inflict a loss of −1 to player A. Similarly, on choice of rock *r* by player A, player B will choose paper *p* to inflict a loss of −1 to player A. However, the game is played in simultaneous format, where on occurrence of certain event (such as countdown), each player brings the hand hidden in the back with any one gesture of the rock paper or scissors. This infuses randomness in the game, making it probabilistic. Usually, the game is played for several cycles. A player may choose a fixed strategy or a probabilistic strategy. When the same set of players play the game, then over a period of time, they tend to know the behaviour of each other. Though each move is independent, their decision may be influenced by the earlier occurrences. After several cycles of game, each player tends to learn about the behaviour of the opponent, and hence, previous moves may affect the decisions of each party.

15.5.3 Some Popular Computer Games

Game theory has been used in several computer programs to achieve a level of strategy execution comparable to that of human beings. Computer programs have been developed to play games such as backgammon, chess, draughts, bridge, scrabble, etc. It may be noted that such capability may reflect power of machine and software in defeating a human being, but in no way, it can be termed as evidence of machine intelligence, though it encompasses decision-making of very high order. Success in Turing tests would be more appropriate when it comes to the comparison of machine intelligence.

Chess

A chess playing computer program Deep Blue developed by IBM generated ripples across the world in 1997, when it defeated the then chess world grandmaster champion, Garry Kasparov. It was seen as a major achievement by any computing system in the field of artificial intelligence. Deep Blue carried out a parallel search of a chess tree. The machine has a capability of performing evaluation at the rate of nearly 200 million moves per second. Deep Blue uses α−β search with further heuristics such as singular extensions which further reduces the search space. Deep Blue was fine tuned to match with grandmaster's games.

Draughts/Checkers

The game of draughts/checkers has been automated to such an extent that it is difficult to

beat the machine. Chinook was developed in 1989 by a team comprising Rob Lake, Paul Lu, Martin Bryant, and Norman Treloar led by Jonathan Schaeffer. Around 2007, it was announced that the program has mastered the game to perfection and cannot lose a game of checkers. The program uses a library of opening moves from games played by checker's grandmasters, a deep search algorithm, a move evaluation function, and terminal game database for all positions with eight or less pieces.

Go

The game is played between two players, who alternately place black and white pieces in an empty cell on a grid. The size of grid may vary. Once placed on the board, pieces cannot be moved elsewhere, unless they are surrounded and captured by the opponent's pieces. Go represents a zero-sum, perfect information, partisan, deterministic strategy game. Computer programs have been developed successfully for small board size. For large board size, the complexity becomes high and results in prohibitive search space.

SUMMARY

Aspects of game theory have introduced possibility of intelligence, planning and decision-making in machine. Various concepts have been used to enable computers play games, which were otherwise thought to be possible only by entity with intelligence level comparable to human beings. Using the concepts of game theory, several problems in real life can be visualised or solved.

Depending on the nature of game, it may fall in one or more categories such as symmetric game, zero-sum game, perfect information game, simultaneous game, sequential game, repeated game, signalling game, large poisson game, non-transitive game, global games, etc.

Players or teams in a game may have several game strategies. Some of the popular game playing strategies are dominant strategies, pure strategy, mixed strategy, tit for tat, collusion, backward induction, etc.

Strategy adopted by a participant may attain the level of stabilisation. Depending upon the state of strategies used by a player or team, a level of equilibrium may be attained. Based upon the nature of game and player strategies, several game equilibriums have been proposed—Nash equilibrium, Sub-game Perfection, Bayesian Nash, Perfect Bayesian, trembling hand, proper equilibrium, correlated equilibrium, sequential equilibrium, Pareto efficiency, self-confirming equilibrium, etc.

Games can be mapped as a move search problem to optimise the gain with respect to a team or player. Minimax approach is one popular approach to find such solution. In any reasonable gaming problem, the dimensionality of search space can rise exponentially. Alpha-beta pruning helps in reducing the search space.

KEYWORDS

1. **Alpha-beta pruning:** This returns same optimal value as returned by the minimax, but is efficient in termination because it reduces the search space.
2. **Game:** In a game, multiple (two or more) players or teams or adversaries or agents participate with the conflicting goals, leading to search and optimisation problems taking into consideration various participating entities. The aim of search is to find a move with respect to a participant in order to optimise its gain.
3. **Game components:** A game consists of initial state (which gives initial board position), permissible moves (a set of rules in a given state), states (reflecting progress of game at any instant), evaluation function [which gives list of pairs containing (moves, state) from any state], terminal condition (reflecting end of game with payoff awards such as won, lost or drawn or any other value).
4. **Game equilibrium:** It is a state of game when participants stick to their strategies for optimal results.
5. **Game theory**: It is a branch of Mathematics highly influenced by economist, which deals with strategies of finding optimal moves for the participants.
6. **Game tree:** A data structure to capture opponents is the game tree. It gets alternate player at the next level.
7. **Imperfect information games:** These are the games in which adversaries may not be aware of the moves made by opponents such as card games.
8. **Minimax algorithm:** It deals with finding optimal value for gain of loss of the player at root (which is mostly the maximising player's ply).
9. **Nash equilibrium:** It is a strategic profile in which no player has an incentive to deviate from the specified strategy.
10. **Payoff matrix:** This matrix captures reward to each player for each combination of actions by all the players. Players may or may not have different sets of legal moves.
11. **Perfect information games:** These are the games in which adversaries are aware of the moves made till the current state by each players at any instant such as chess.
12. **Zero-sum game:** In this game, a participant's gain or loss is balanced by losses or gains of the other participant(s).

MULTIPLE CHOICE QUESTIONS

CHAPTER 11

Expert Systems

Learning Objectives

- To understand the relationship between an expert system and AI
- To understand the concept of expert system and the process of building the same
- To study the importance of knowledge representation for expert systems
- To study the role of reasoning
- To identify the impact of uncertainty in expert systems
- To gain knowledge about inferring and various types of expert systems in practice

INTRODUCTION

Consider the bidding process of Indian Premier League. The owners of participating teams look upon the experts to assist them in the selection of a player during the bidding process. Similarly, a newly joined employee in a company looks upon a financial adviser to give him an idea of investments and so on. So, everyone in some or the other way relies on the expert to get some sort of recommendations, suggestions or guidelines to assist in taking decisions.

So, what is an expert system? An *expert system,* as the name suggests, is a computer system that possesses or emulates all the decision-making capabilities like that of human expert. When we say so, do we mean to say that any program can be called an expert system when it has these capabilities? Almost all the problems discussed so far have been exploited in some or other expert system. What matters the most is the representation and the mapping mechanism of the knowledge with a strong reasoning and inferring process. Now, you would feel we are going back to the chapter of knowledge representation, and yes, you are right; it affects the overall mechanism of an expert system.

Expert systems (ES) have their roots in cognitive science, where the study of human mind exists with the combination of artificial intelligence and psychology. Expert systems are said to be the first successful applications of AI to have real-world problem solving. A few of the areas of its applications are medicine, finance, space research and so on.

Let us go through the history of ES. A rule-based system was the first one that was proposed by E. Post in 1943. Its typical example is a system based on the if-then rules. Then in 1961, a General Problem Solver was proposed by A. Newell and H. Simon. Later, it was the DENDRAL and MYCIN, which are the most representative examples of ES studied till today. These systems are described further in detail.

Precisely, we can say that an expert system is the one that finds its place in problem solving domains and gives a reasonable solution. Thus, it comprises knowledge base and a strong inferring mechanism. It is a system that is capable to reason out the actions and the outcomes it suggests. Hence, the ES can be classified into the areas of application for prediction, planning, monitoring, diagnosis and so on. Let us begin with the architecture of expert system.

11.1 ARCHITECTURE OF EXPERT SYSTEM

With reference to our introduction, we are already familiar that an expert system needs to have a strong knowledge base along with an inference mechanism. Figure 11.1 depicts the ES in operation.

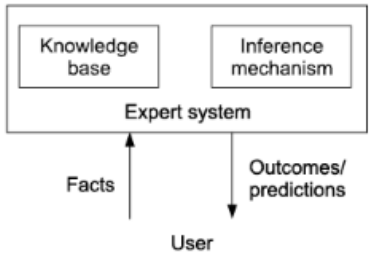


Figure 11.1 Expert system: Working.

What forms the knowledge base (KB) for the ES is the knowledge that it has attained from humans or some sort of information that may be in text or other form. This forms the factual knowledge—the knowledge that is generally agreed upon. The other type of

Return to page 391

knowledge that KB can contain is the heuristic—the knowledge that is based on experience and that helps in good predictions. The inference mechanism is responsible for getting the conclusions with the help of KB. The user tells the facts and asks the expert to give a solution. The ES accordingly processes the rules and gives the outcomes.

Further, in an expert system, we can relate two types of knowledge—one is problem knowledge that is specific to the type of problem at hand and the other is domain knowledge, where an expert knowledge persists with the know-how about solving specific problems. So, the domain knowledge forms the subset of problem knowledge. Hence, both of them are influential in the formulation of KB. This knowledge can be represented in many ways.

How can the expert system be designed? There is need to convert knowledge so that it can be initialised by the expert system. Building the KB is a two-phase process. In the first phase, the knowledge engineer sets up communication with the human expert. This is essential to get the maximum of the knowledge that can help in inferring. In the next phase, the knowledge engineer codes in the knowledge for KB explicitly. The evaluation for such a system is then again carried out by the expert. The expert here acts as a critic whose inputs would help in refining and building a better system. Figure 11.2 shows the development process.

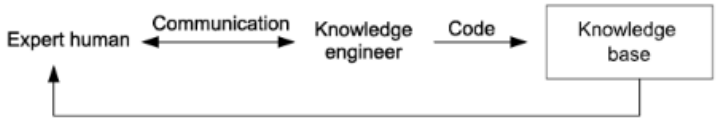


Figure 11.2 ES: Building process.

11.1.1 Parameters in Building an Expert System

After describing the architecture of the ES, we move to the basic parameters in building the system. The parameters described here essentially tell the activities required to be undertaken during the building process. The three basic parameters are described here briefly, viz. *domain exploration, expertise transfer* and *meta knowledge.*

Domain Exploration

Domain exploration is concerned with the exploration of the knowledge about the domain in which the system is operational. This knowledge is valid and can be directly applied and made use of. Various experts or specialists are in picture for this domain exploration. Here, the objects and their relationships to the domain are identified along with the other relevant information essential from the KB building perspective.

Expertise Transfer

Expertise transfer is the transfer of expertise from one system to another. In ES, this expertise transfer takes place from acquiring the knowledge from the expert to the transfer of it to the user. This takes place during data acquisition phase. Once the knowledge is obtained, there is a transfer to represent it for the system to understand. At later stage, the transfer takes place to build an interface and finally, to have the outcomes transferred back to the user. So, at every stage, it is understood that some expertise is transferred. Figure 11.3 shows the stages, where the expertise is transferred in the form of knowledge.

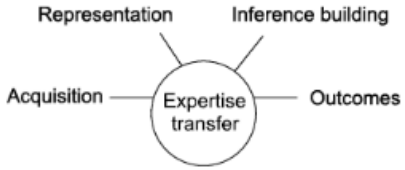


Figure 11.3 Expertise knowledge transfer stages.

Meta-knowledge

We know that *meta-knowledge* is a knowledge about available knowledge. But how is it used as a parameter block for building an expert system? We have already discussed about domain knowledge and problem knowledge. Meta-knowledge is systematic as well as domain independent knowledge. This actually helps in performing or building an expert system that is less dependent on a particular domain, thereby helping in keeping a generalised view. Thus, the meta-knowledge actually illustrates everything about the domain. Figure 11.4 shows where the meta-knowledge lies in the hierarchy structure.

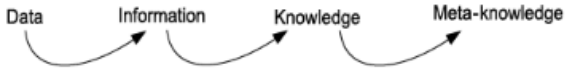


Figure 11.4 Meta-knowledge: Hierarchy structure.

11.2 CONFIDENCE FACTORS

There is a notion of confidence factors in expert system. This is like an expert suggesting a student some college to join, but at the same time, gives another option of some specific course not offered in that college that could be beneficial. So, there is some uncertainty in this scenario because at the same time, the expert gives multiple options on the basis of some confidence factors. *Confidence factors* are the values associated with a particular variable. They indicate the degree of truth for that particular variable. So, there is a fuzzy notion here.

Obtaining the Confidence Factor

Based on the confidence factor, an expert system can give guidelines or

recommendations. But from where can we obtain this factor? It can be made available in various ways, which are as follows:

1. ***User input:*** A user specifies the confidence level for some specific variable. This could be direct or indirect. For direct, let us take an example. Suppose there is a question like 'Do you plan to appear for IIT entrance exam?' and a possible answer from the user is 'Yes, 50% chances are there that I will appear for the exam.' Here, this 50% provided by the user is the confidence factor. In indirect form, a database is used by ES to determine confidence factor. This is based on the inputs obtained from the user. Consider a scenario in which the ES is used to give recommendations for the selection of a mobile. Let us assume that for different types of mobiles, a database containing rankings obtained from user is maintained. These rankings act as the confidence factors for the ES to give suggestions.

2. ***Rule set:*** Other approach of having the confidence factor obtained is simply the use of rules. Assume a rule 'if (degree=Computer Engineering) and (Masters=Computer Engineering) then High_salary_job=true' with a confidence factor of 75. Such rules associated with confidence factors can be based on previously available information and heuristics. Further, these rules can propagate these confidence factors to the subsequent rules.

Moreover, probabilistic approaches can assist in developing these confidence factors based on the previous available probable values for any variable. A simple probability value can be treated as the confidence factor. Yet in many cases, we cannot say that the probability value can prove to be effective in giving judgments, as there might be next level testing requirement, where the earlier value fails.

11.3 EXISTING EXPERT SYSTEMS

In this section, we discuss early expert systems—MYCIN and DENDRAL.

MYCIN

MYCIN is an early expert system developed in 1970's at Stanford University to provide users with an explanation of its reasoning. It was developed to aid the physicians in diagnosis and treatment of meningitis and bacterial infections. So, it used to provide a consultative advice about the infections. The diseases handled by MYCIN are observed to be developed during a recovery from some other disease. Physicians need to act fast in case of such complications. Taking samples of the patient and getting results from laboratories generally takes too much time. It is essential that the treatment should be started prior; even though the entire laboratory results are not available. Under this scenario, MYCIN is useful.

MYCIN is a computer program that takes inputs form the physician and gives advice. For this purpose, it invokes a dialogue. The physician answers the questions and on the basis of the knowledge MYCIN has, it provides diagnosis and detailed drug recommendations. It reasons about the data. It also takes into account the characteristics of the patient and the available laboratory results. Working in two phases, it performs the diagnosis in the first phase and drug recommendation in the second phase. MYCIN also considers the certainty factor based on the doctors confidence that assist in decisions.

Reasoning being a very important characteristic of the expert system, the physician can ask 'why' in between the process. At this instance, MYCIN explains the hypothesis that is under consideration. At the end, it is also possible to trace the entire reasoning process.

A snapshot (hypothetical) of the dialogue between MYCIN and the physician is given below:

The initial input consists of the following:

Patient's name: Mohan
Age: 30
Sex: Male

Q: Which type of test is it?
A: Blood
Q: How many days ago the blood test was obtained?
A: 3

Here, if any of the questions asked are not known, then the physician can type 'unknown'.

Q: Was Mohan a diabetic patient?
A: Why?

At this instance, the hypothesis is stated by MYCIN. For example, MYCIN puts forth the following hypothesis:

It is known that the test is of blood. The platelets count is on higher side. Then, with the rule, if the platelets of the patient are on higher side and the patient is diabetic, then with certainty of 0.5, identity of the disease is thrombocytosis.

From such a dialogue, it makes some prescriptions; a sample is given below:

1. XYZ medicine
   Dose: 770 mg (calculated on the basis of 15 mg/kg)
   Comments: Repeat blood test after 2 weeks.
2. ..... and so on.

The above discussion gives an overview of the system. But it is obvious that the entire accuracy is based on the rules. For every input provided, MYCIN takes decisions from the set rules with reference to the certainty factor and further proceeds to the next action. For example, it can have rules that conclude likelihood of the infection. For every rule, a

unique number is applied. So, rule 333 could have 0.8 likelihood for some disease with an answer, yes, the infection is likely to occur' or in some other rule, say 366 could have 0.45 likelihood with an answer 'no', indicating the disease is less likely to occur.

Thus, MYCIN has displayed the use of expert system in medical domain. The expert systems developed after MYCIN have actually set MYCIN as a benchmark. Accuracy of MYCIN is based on various factors. One of them is it assumes that the physicians are familiar with the terms it makes use of. Other factor is even though it has showed great ability and usefulness in its applications, its operation and reasoning process are restricted with the available knowledge it has. A physician cannot ask 'why' for a particular rule. So, the existing rules cannot be questioned.

To summarise, MYCIN represents the first of the new generation expert systems having an ability to reason. MYCIN's evolvement has actually set up a trademark in the research field of AI. Even today also, it is used for other research purposes with some modifications. Allowing rules, to be added incrementally and using them further have made the expert system a strong one, helping for better justifications and decisions.

DENDRAL

AI researcher, Edward Feigenbaum and geneticist, Joshua Ledeberg started work on heuristic DENDRAL in 1965. It was later called DENDRAL. The intention of this work was to have an expert system for chemical analysis. DENDRAL analyses the organic compounds so as to determine their structure. This was one of the early examples of the AI program.

DENDRAL infers the structures of the organic compounds. For this, a strategy, called *plan-generate-test* is used. The strategy comprises three sub-programs—the generator, the planning program and the testing and ranking programs. The generator forms the core of the heuristic approach. It comes out with potential solutions that are further generated into larger graphs of chemical atoms. In order to restrict the structure generation, CONGEN technique is employed that further assists the chemist in determining the chemical structure. Even though CONGEN has made things simple and is a standalone package, further assistance in inferring occurs with the planning programs. A large amount of knowledge is put to use in planning programs. The planning programs analyse the data and infer the constraints. It could come out with solutions revealing that an unknown molecule is a ketone, but not a methylketone. The planning information is actually provided with the list of good or bad features to the generator module. The last module actually ranks them. Programs like MSPRUNE and MSRANK are used here for making testable predictions from the plausible candidate module. This includes the analysis of the solution and discarding the scenarios that are not satisfying specific criteria.

To summarise, the stages of DENDRAL are initial stage of planning and to have hypothesis formation that helps the generator to have potential solutions and further tester to have predictions.

DENDRAL programs have been very much in practice for the problems related to structure determination of organic acids in human urine, antibiotics, and impurities in manufactured chemicals and so on.

One of the major problems faced by DENDRAL was the extraction of domain specific rules from experts. To overcome the same, Meta-DENDRAL was initiated. It is an automatic rule formulation program that has been structured for building a strong knowledge base. This has further resulted in higher accuracy in the prediction of correct spectra of new molecules.

With the success of MYCIN and DENDRAL, there has been a rapid growth in the development of new and better expert systems. At present, many expert systems have been developed with the baseline of these existing ones.

11.4 KNOWLEDGE ACQUISITION

Generally, in expert system, the knowledge engineer takes inputs from the domain expert to build the KB. Things get more complex when the expert is unwilling to co-operate to give in the fullest information. Further, the KB that is built needs to be refined so as to match the required performance. The process of acquiring expert knowledge, hence, becomes a complex task. Thus, to construct a good KB, acquisition has to be at its best.

Most often, a set of programs is compiled to extract the expert knowledge effectively. These programs called *knowledge acquisition systems*, though not fully automatic, assist in building effective KB. These systems provide support for different activities and are most effective when confined to specific problem-solving task. The tasks they carry out are input of the knowledge, maintaining the consistency and finally, making it richer. With respect to the specificity of the given problem for which the expert system is to be designed, the acquisition process should work in that direction. As an example, if we are to design a system that is meant for diagnosis purpose only, then the knowledge acquisition should be concentrated more on the hypothesis and the possible cause-and-effects. This helps in the identification of different causes for one hypothesis. Thus, the acquisition system is able to get maximum benefit by getting inputs on unknown causes (more knowledge) from the expert. Moreover, the decisions to be taken in case of conflict are resolved when appropriate inputs are made available from the expert.

There are various knowledge acquisition systems, viz., MORE, MOLE, SALT, LEAP, OPAL, TIMM, Meta-DENDRAL and so on. We discuss a few of them.

MORE is basically a tool that assists in eliciting knowledge from the experts and is most commonly used for diagnostic inferring. It refines the domain model with newly available information that can form hypothesis. Once a rule set is developed, the expert is asked to put in more knowledge for better rules set development. It makes use of qualitative model of causal relations to have the diagnosis.

MOLE is an extension of MORE. MOLE helps the experts in building a heuristic problem solver. It has the capability to disambiguate the under-specified knowledge and refine the KB that is incomplete. MOLE develops a KB that is called *MOLE-*

p(performance). It carries out various tasks that include formation of the initial KB and its refinement. It begins with getting an initial knowledge regarding the symptoms and the other details from the expert. It prompts the expert for explanations of the symptoms till it comes to the precise and ultimate set of causes. Determining correct explanations in case of conflict is also resolved by MOLE. Inferring anticipatory knowledge is also handled. In refinement stage, it identifies the weak slots and tries to fill them with more explanations from the experts. Finally, it is able to extract the knowledge intelligently to build a reasonable KB. MOLE has been used for different diagnostic purposes like diagnostic problems in car engines and others.

SALT is a knowledge acquisition system that works on the principle of propose and revise. This is useful for design tasks. SALT was developed for an elevator system configurer. It incrementally constructs designs by proposing values for design parameters. At the same time, constraints are noted and in case of violation, domain expert is used to revise the past decisions. In the beginning of knowledge acquisition, the user is provided with a menu, which operates for three different types of cases—procedures for proposing values, constraint identification on individual parameter values, and suggestions to revise the design, if the constraints are not met. Once this information is available, SALT stores the knowledge in dependency network.

Despite the tremendous growth in different knowledge acquisition techniques, there is still a necessity to have efficient ways for acquisition for more efficient expert systems.

11.5 SHELLS AND EXPLANATIONS

Shells are the interpreters that are used for building the ES, whereas *explanations* allow the system to express itself when it applies some reasoning. Let us discuss in detail.

Shells

Imagine you have to develop an expert system—a system that has to perform some predictions with regard to the weather. First and foremost, you would collect the information. Rules would be encoded with the formation of KB. Then, you would have an interpreter for it. Here, the data is all about the weather conditions and related information. Now, if you have to develop a new system for predicting traffic conditions, you would start from scratch. Instead if you have a shell, then by adding just the problem knowledge, things would get solved. So, as said earlier, for the ES, where the rules and the interpreter are combined together, the interpreter can be separated from the domain-specific knowledge, enabling to build new systems. These interpreters are shells. Sometimes, they are also called *AI tools or skeletons*.

EMYCIN (empty MYCIN) is an example of such shell. Building the shell has many advantages and one of the biggest one is that is simple and you need to enter only task knowledge. The inference mechanism that exploits the knowledge is built into the shell. Further, if the problems are not very complicated, the expert too can have the knowledge entered in. Figure 11.5 shows the shell structure.

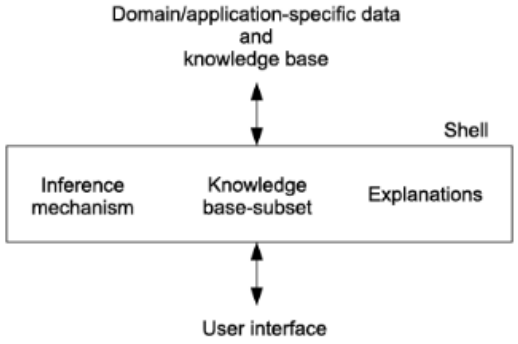


Figure 11.5 Shell structure.

At present, a large number of shells are available in the market. They tend to give a better flexibility in terms of knowledge representations and reasoning with the support of rules or frames or other mechanisms. Some shells also have mechanisms for the representation of knowledge as well as for reasoning and explanations. But remember though shells help in the building systems with simplicity, they do not help in knowledge acquisition. But to mention—some shells also have tools that help in better knowledge acquisition. Finally, the power of an expert system lies in the knowledge; more the knowledge, more will be its efficiency.

Explanations

Explanation is an important feature of the ES, which shows the ability of the system to express itself. In general, there is a necessity to have explanations for any action that is being taken. We do not accept things unless they are clarified and we are convinced about them. Similarly, for making the ES effective, it should be interactive. By this, we mean that the explanations as to how a particular conclusion has been drawn upon should be made available. For example, suppose an ES suggests a course to a student; it should be able to explain why that course is beneficial to him/her.

The explanations are based on the rules that are used for the inference process. So, these rules are the ones that help in reasoning. They are very important from the knowledge engineer perspective too, as they give the path in which the inference has been drawn.

Are the explanations really required? Depending on the system, say in case of bird identification system, the explanations could just be obvious, but in some cases, they are really vital, say in case of medical diagnosis.

Return to page 391

In the course of explanations, the ES asks you some questions to infer. The question that comes to your mind is 'why?'; why is there a need for the system to ask? To arrive at a conclusion based on a particular rule, it asks the questions. With the backward reasoning and the previously answered questions, the ES is capable of giving these explanations as well.

So, we can say that two things are covered in explanations. One is when the ES is asking you questions, you need an answer as to why these questions are asked, and second is when a conclusion is derived, then an answer is needed as to how it has been drawn?

11.6 SELF-EXPLAINING SYSTEM

Self-explaining system is a feature of the expert system that shows its ability to explain itself. Self-explanation means that it is able to justify why a particular question is being asked and how it has arrived to the conclusions. We have just discussed about the explanations and it precisely specifies the self-explaining system. In some systems, there could be a self-explaining module inbuilt, typically in medical diagnosis, where there is a requirement of reasoning for diagnosis purpose.

11.7 RULE-BASED EXPERT SYSTEMS

Rule-based systems consist of rules (if-then), set of facts and an interpreter. The rules are the ones that constitute the complete KB. Rule-based systems are essentially designed so that they can have heuristic reasoning and proper explanations. They can separate the knowledge from the reasoning and facts. Figure 11.6 depicts a rule-based system. The inference mechanism, often referred to as *inference engine,* seeks the necessary information and the relationships from the KB to answer. It is also responsible to find the right facts. Two types of inference mechanisms are used—forward chaining and backward chaining. Explanations make the user understand how a particular conclusion has been derived. So, they trace the rules in the inference chain and provide user with the facts/rules used in the process.

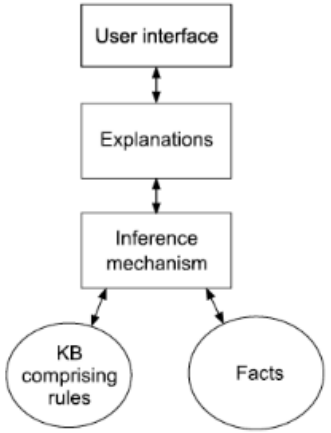


Figure 11.6 Rule-based expert system.

A simple example of if-then rule for a rule-based system that is in the form of if antecedent, then consequent, is given below:

If the number is divisible by 2, then it is even.

This is a very simple rule, but the rules can be complex, where multiple antecedents and consequents can exist. Multiple antecedents are often connected with 'and' or 'or', i.e., conjunctions or disjunctions. For example, if the students are absent or the students have not done the submission, then there will be no class.

Even there can be multiple consequents, which can be connected with conjunctions.

From the discussed example, it can be noticed that the rules fall under certain semantics. The semantics tells us that the production rules have some meanings. The semantics can also be overlapping rather that explicitly belonging to a particular category. They are broadly classified as follows:

1. ***Relation:*** Here, the relationship between the antecedent and the consequent is specified. For example, if the water is boiling, then the water is hot.

2. ***Recommendation:*** A rule that directs or gives some advice. For example, if the water in glass is hot, then advice not to touch the glass.

3. ***Strategy:*** A strategy tells what needs to be checked. These types of semantics advice regarding actions for betterment. For example, if road 1 is blocked, then try road 2.

4. ***Directive:*** It refers to an implication or indication of some action that needs to be

carried out. It is a part of directive semantics. For example, if the light is green, then you can proceed.

5. ***Heuristic:*** It states guidelines-based rule. For example, if there is severe pain in chest and difficult to breathe, then the diagnosis might be as heart attack.

With the above discussion on the rule-based expert systems, we can say it is just matching of the rule to come to a conclusion, but there can be cases of conflicts also. Conflict can arise with regard to the selection of rule, or there could be conflicts with respect to the consequent mapping too. There can be different approaches to handle this. One way is to apply explicit ranking to the rules. This means prioritising them. Another approach can be to have a sequential way to trigger the rule. Firing of the rule can also occur on the basis of the current data that is treated having higher ranking or is found to be potential. Sometimes, an approach of triggering the longest rule is also taken up in case of conflict.

Let us proceed towards the pros and cons of the rule-based expert system. The major advantage of a rule-based expert system is the way the knowledge is represented. It resembles the human reasoning process, and hence, becomes effective. It can handle uncertain knowledge as well. When the ES deals with huge amount of data, managing and updating the KB get complex. Further, the conflict resolution gets more complicated with the growing size. Another problem is the rigid behaviour of the system that makes it useless outside the scope of the domain.

11.8 FORWARD AND BACKWARD CHAINING

In the rule-based expert systems, we have mentioned about the inference mechanisms that were used in the processes, viz. forward chaining and backward chaining. In forward chaining, as the name suggests, the facts are processed first. It keeps on looking out for the next rule till it is fired and proceeds. So, a rule is fired where the antecedent is matched, else it halts. Forward chaining is also called *data-driven approach*. Let us take an example that has the following rules:

Rule 1: If *P* and *Q*, then *A*.
Rule 2: If *P* and *R*, then *B*.
Rule 3: If *S*, then *R*.
Rule 4: If *B*, then *C*.

To prove: *C* is true given the facts that *P* and *S* are true.

Here, the forward chaining will work in the same way as explained above. It starts from the first rule till a match is found. The antecedent match is found at rule 3. From this, it is concluded that *R* is also true. So, addition of a new fact states that we have *R*, *P* and *S* as true. The next rule that is fired is rule 2 (as *P* and *R* are true). This adds that *B* is also true. This helps in triggering rule 4, where *C* is true; hence it is proved.

Considering the same example for backward chaining, it begins in the reverse way. The approach is commonly referred to as *goal-driven*. Here, from the goal/conclusion, it moves to find out evidence to prove it. (The way police traces the culprit once any crime has occurred.) Given to prove *C* is true, it will now proceed for matching the consequent rather than the antecedent. Starting from rule 4, it keeps moving till the given facts are found to be true. The antecedents are now made new sub-goals.

A question that comes to our mind—what should be used and when? Forward chaining is used when data collection is an expensive task and the size is limited. Backward chaining is useful when the data to be handled is substantially large. It is also a common practice to use backward chaining, where specific characteristics are looked upon by the expert system.

11.9 FRAME-BASED EXPERT SYSTEMS

With the study of rule-based expert systems and the forward and backward chaining methods, let us begin with the frame-based systems. The expert systems which make use of frames for the knowledge are called *frame-based expert systems*.

But what is a frame? A *frame* is a data structure with a typical knowledge about the object or the concept. We are already familiar with Minsky's frame. So, a frame has its name and a set of attributes. For example, a car frame can have make, type, colour and so on as the slots/attributes in the frame. Each of the slot or the attribute has a unique value associated with it. So, we can say that the frame gives a concise representation. Hence, a frame-based representation is suited for object-oriented programming techniques.

In AI terms, a knowledge engineer actually calls the object as the frame. The slots can contain a default value, or a pointer to another frame, or a set of rules or procedures so as to obtain the slot value. To detail it more, we can have the following things included in the slot:

1. Frame name
2. Relationship with other frames
3. Values or ranges
4. Procedural information

By relationship, we mean that the hierarchy, or to be specific, the inheritance is depicted. Value and ranges, as discussed earlier, represent the actual/default values or the specified ranges. In case of procedural information, the slot is attached to a procedure that is executed when any event is triggered such as change in the value of the slot. Generally, an instance-frame refers to an object and a class-frame refers to a group. So, car, person, bird can be referred to as class-frames. Figure 11.7 represents the frame structure for class and instance.

Return to page 391

Class: Car
Make:
Type:
Colour:
Airbags:
Auto-transmission:
Windows:

Class: Car Instance: i10
Make: Hyundai
Type: Small car
Colour : Red
Airbags: 2
Auto-transmission: No
Windows: Power

Class: Car Instance: Verna
Make: Hyundai
Type: Sedan
Colour: Blue
Airbags: 4
Auto-transmission: No
Windows: Power

Figure 11.7 Frames: Class and instance.

Since we have been discussing that frames are most useful in object-oriented approach, the generalisation and aggregation concepts apply here as well. For example, engine and chassis can be the parts of a class car. Similarly, car is a kind of a vehicle. So, both the concepts are applicable here. Can there be any association? Yes. An association can exist between the classes, say a book class may belong to a person or in other words, the book is owned by a person.

Till now, we have concentrated only on the knowledge part, but as we know that an expert system needs to have some methods too, so, methods and demons are added to the frames. A *method* is a procedure that is executed when requested. It is associated with the frame. Sometimes, we want a specific event or an action to occur when any value changes. Based on these facts, the methods are categorised into two categories when changed, and when needed.

A *when changed method* is executed when the value changes, whereas a *when needed method* is used to obtain the value for the attribute. This happens in course of problem solving that the value is required. We have discussed about the demons too. What is the difference between a method and demon? A demon makes use of if-then structure, whereas methods are used for more complex things.

Working of a Frame-based Expert System

After having understood the concept of frames, slots, methods, let us study its working. We know that in rule-based expert system, it is the chaining mechanism in the inference that fires the rules so as to reach the goal. In case of frames that have the knowledge representation, the methods or the demons essentially add actions to them. The goal, hence, is set in the methods or the demons.

For example, the process for an expert system to check the eligibility of a student appearing for an exam is explained (refer to Figure 11.8). We have a set of slots/attributes as well as methods and demons for the frames of check eligibility, request, actions and so on. A when changed method is invoked when the expert system is asked for the evaluation status, whereas when needed method is, in turn, invoked by the system to check the submission status of that particular student. So, there can be a flow from a staff to get the details of the student for evaluation. Let us assume that this frame has an attribute, i.e., evaluate eligibility. A sample view of the same is depicted in Figure 11.8.

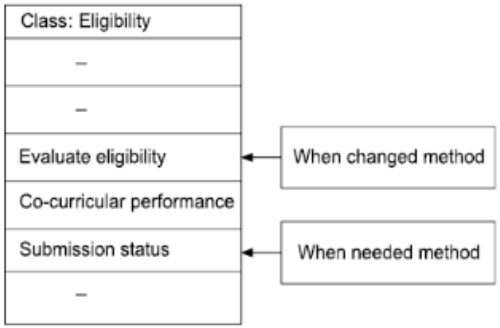


Figure 11.8 When needed and when required methods: Snapshot.

Guidelines to Build a Frame-based Expert System

1. The first step that is involved in designing any expert system is the scope and the specificity of the problem at hand. There has to be utmost clarity about these two factors.

2. The next step is the identification of classes, instances and the attributes.
3. Since the events or to be specific, the methods are the key role players in evaluation, the display of the system has to be presented in the most simple and transparent manner.
4. The next task is to have the methods of when changed and when needed. This is dependent to the design in step 3, where the events are decided.
5. Rules are to be also defined along with the methods.
6. Finally, a full-proof evaluation of the built system, and if required, expansion should be handled.

A detailed example is handled in the case study for the frame-based expert system, provided further in the chapter.

11.10 FUZZY EXPERT SYSTEMS

Many things are uncertain and most often, we face a dilemma with respect to determination of belongingness of a particular thing. Still, as humans, we are in position to handle the reasoning in such cases. A *fuzzy logic* is the one that provides mapping for this uncertainty. To put it in simple words, a *fuzzy expert system* is the one that makes use of fuzzy membership functions. In fuzzy logic, as we are aware, approximation exists rather than exactness. The same concept is put to practice in these expert systems.

The most common rule that is found in the fuzzy expert system is in the form of If X is low and Y is high, then Z is medium.

Here, X, Y are the inputs and Z is the output. Z is the one whose value is to be computed. Low, high and medium define the degree of membership. The expert systems based on fuzzy can have multiple conclusions per rule. So, the set of fuzzy rules comprises its KB.

Figure 11.9 depicts the fuzzy expert system. Every expert system has an inference mechanism, but the way in which the mechanism derives the conclusions is different. In this case, the four processes that are commonly used are as follows:

1. Fuzzification
2. Inference
3. Composition
4. Defuzzification

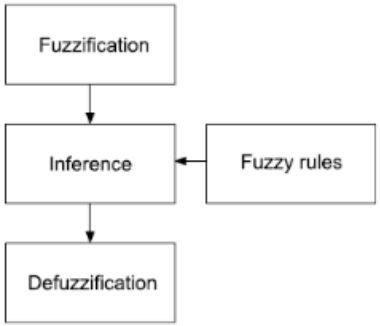


Figure 11.9 Basic architecture of fuzzy expert system.

In *fuzzification* process, the membership functions are defined on the variables. Input variables are applied to the actual values. This helps in determining the degree of truth for each rule premise. This degree of truth for the premise is called as *alpha*.

In the *inference process*, the truth value for the premise is calculated. The inference mechanisms that are applied are min and product. In case of *min*, the output membership is cut off at a specific height that maps to the computed degree of truth of the premise. In *product*, the output is scaled by the computed degree of truth of the premise.

In *composition process*, the fuzzy subsets are combined to form a single fuzzy subset for output variable. Composition process consists of max and sum compositions. In *sum* compositions, point-wise sum over the fuzzy sets is considered. Whereas, in case of *max* composition, point-wise maximum value over the fuzzy sets is taken.

In *defuzzification process*, the fuzzy value is converted to a crisp value. The most common methods are centroid and maximum methods. In case of centroid approach, a fuzzy centroid is calculated, while in *maximum* methods, average-maximum is the one that returns average value where maximum truth value occurs. (Detailed example is covered in Chapter 18.)

Fuzzy expert systems are most effectively used in the category of pattern recognition. There are many advantages of fuzzy systems. To mention a few, the KB has limited number of rules, and hence, is efficient. Since the knowledge is structurally shallow, the run-time chaining for the inferences are not required. They exhibit less maintenance costs and can be built at a faster rate. The KB is semantically rich, and hence, the fuzzy experts systems can be well-suited for decision-making.

11.11 UNCERTAINTY MANAGEMENT IN EXPERT SYSTEMS

Uncertainty exists in every aspect. It means lack of exact knowledge that allows us to

come to an exact conclusion. So, can we say that the information we have is partial? Yes, the information can be incomplete and uncertain.

There are many reasons of this uncertain knowledge. Even in writing, we use the terms —often, hardly, may be, sometimes. While using these terms, we are uncertain about some factors. Uncertainty can arise when we get different opinions possibly from different sources. Even in some cases, the data values are actually missing too. For expert systems, to build concrete rules, the dependency on the domain expert adds up more complications.

So, the experts systems need to have probabilistic approach for handling the uncertainty. The KB is also treated to be a major source of uncertain knowledge. Rule-based expert systems are good at handling of the uncertain knowledge. Though numerous methods are worked on for the uncertainty management (certainty factors, Bayesian probability, Dempster–Shafer theory are the ones that are most often used), yet the selection of method is dependent on the available reliable statistical information for the application.

11.12 EXPERT SYSTEM AND DECISION SUPPORT SYSTEM (DSS)

What is the relation between ES and DSS? Till now, we have discussed about ES which assists in taking decisions. The fact is that the ES is used in DSS. So, it is basically an expert system technology that is embedded in DSS. On a broader scale, DSS is categorised into data-oriented DSS, model-oriented DSS and process-oriented DSS. Data-oriented DSS focusses on data warehouse, model-oriented DSS handles small databases, whereas process-oriented DSS deals with the question-oriented models. Thus, ES falls under this category of DSS. Though the distinguishing factor between the two is a thin line, we can always say that ES too, of course, will be making the use of these models, where the facts will be made available through the databases.

Often it is observed that ES face the problem of lack of deep knowledge. By this, we mean that in-depth knowledge will always aid in giving improved decisions. Further, system models along with ES can help in providing the support components for computations in the decision process. So, DSS is not built by just ES, but includes other components too that support decisions. DSSs are used at a higher level of business intelligence to avail strategic decisions. DSS, where the data sets are structured, could wholly rely on ES. In case of complex and unstructured data, it uses other methods. Moreover, it is actually the nature of output that is expected from the DSS that will, in turn, determine the way ES is to be used in it.

11.13 PROS AND CONS OF EXPERT SYSTEMS

What can be the possible benefits of having an expert system as machine rather than a human? They are many! A few of them are listed below:

1. A consistent output
2. Quick and fast response
3. Location/date/day/time independent
4. Can be made generalised. (A change in application would result in looking out for a human expert related to that field if we are not relying on expert systems. But with expert systems, having generalisation, the process gets less complicated.)
5. Efficient utilisation of the knowledge (Human experts may forget some aspects, whereas expert systems consider all the rules and scenarios.)
6. Simple future enhancements with additional information of the knowledge and the rules (but with humans, it is difficult)
7. Easy maintenance of system.

The disadvantages of expert systems are mentioned below:

1. They cannot handle new dynamic situation.
2. The systems cannot be adaptive based on the decisions taken earlier.
3. Limited set of knowledge will leave them with limited set of decision outcomes.
4. Development cost could be high depending on the purpose they are used for.

Still, with the listed advantages and disadvantages, whether to have an expert system or not is wholly dependent on the purpose of use.

11.14 CASE STUDY

There has been a substantial growth in the need and development of expert systems for the increasing application domains. With a lot of expert systems to handle the predictions, diagnosis, classifications like that of Buy Smart (a frame-based expert system) that addresses the housing needs or an audit-based system to tackle the compliance norms, there is much to study and discuss. Here, we discuss Prospector—an expert system in geology domain.

Aim of the Expert System

The aim of this expert system is to evaluate the mineral potential of a geological region. It handles the geological settings, structural controls and different types of rocks as well as minerals.

Target Users

The target users for this system are the geologists who are at an early stage of investigating the site to be a drilling site.

Key Points

Prospector can handle uncertain data and incomplete data. It is also a domain independent system. Data is matched to the models to check the presence of ore so as to identify the site to be a drilling site.

KB

The KB of Prospector is divided into two parts—general and special purpose. As discussed earlier, the system is domain independent, so the general KB maintains background information that can be used for other applications as well. The special purpose KB contains knowledge that is relevant to the domain and is in the form of inference network.

Methodology

Prospector makes use of production rules and semantic networks. It makes use of backward chaining inference strategy.

Representation

The representation used by the developers is the inference network. *Inference network* is a collection of evidence and hypothesis. There exists a linkage between the evidences. In short, it explains the factors which actually affect each other. It is in the form of if E, then H, where E is the evidence and H is the hypothesis. The hypothesis has some degree that is pre-stored or ranged. This degree specifies the likelihood of sufficiency and necessity. Also, the values of E and H (probabilities) are changed with the use of Bayes's theorem in the course of execution. So, the inference network actually provides with a way to compute the probabilities of the influential factors. It also makes use of semantic network. It is a network of nodes that is used to represent relevant knowledge like taxonomy relations. It represents the associative memory model.

Inference Mechanism

Since it has to handle uncertainty, probability has to come in picture. It makes use of Bayesian, certainty factors and fuzzy sets. A form of Bayes's theorem that is referred to as *odds-theorem* is used in Prospector. As discussed earlier, there is a measure of sufficiency that is used when the evidence E is known to be existing, whereas measure of necessity is used when E is not known to be existing.

Some of the sample models used in the Prospector are shown below in Figure 11.10.

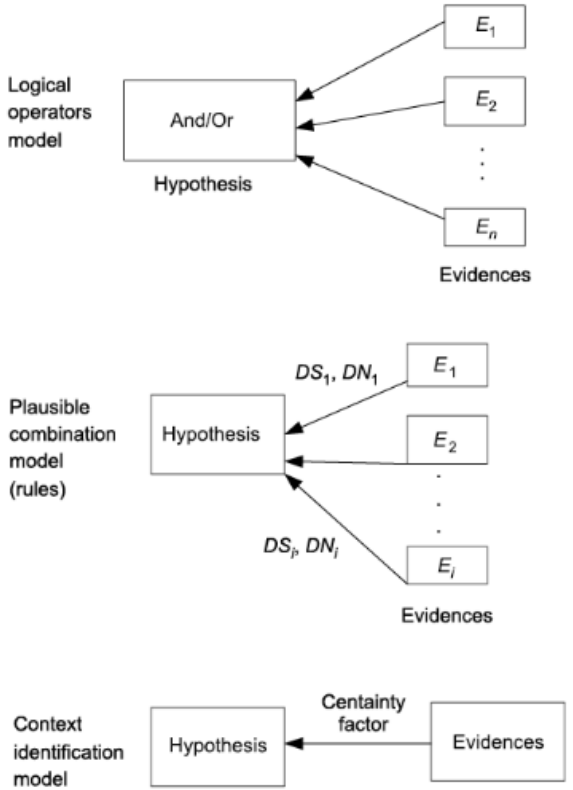


Figure 11.10 Prospector models.

SUMMARY

Expert systems are the ones that help in problem solving. With an efficient expert system, the diagnosis and predictions have become simple. One of the important aspects of expert system is the inference mechanism. Forward and backward chaining are used for inferencing in case of a rule-based expert system. Handling uncertainty is the most critical task and is better tackled by the rule-based expert system. Frame-based expert systems are most suited for object-oriented approach. Fuzzy systems use fuzzy logic and have proved to be better in pattern analysis problems. Expert systems, in a broader sense, are used for prediction, classification, diagnosis, monitoring and control, and many more. MYCIN and DENDRAL are the early expert systems that are now used as benchmark in developing the new ones.

Knowledge acquisition also needs to be taken care of while building an expert system. With the different knowledge acquisition tools that exist, the task of building an expert system becomes simple. The tool communicates and exploits the information that helps in building a better KB. Shells are used further that act as a skeleton for building an expert system. Explanations help in providing information to the user regarding the rule that is applied.

With the tremendous growth in the expert systems, they have proved to be a boon in assisting with better inferences. Still, if you are designing an expert system, make sure that there is a necessity of the system for a problem. Also, be specific with the design and usability of the system.

But before we conclude, we cannot forget the other side of the expert system, the drawbacks! Does the system have common sense to reason about? The answer is no. The system cannot be up to date, and at the same time, is non-incremental in nature. It is not always adaptive and most often domain-specific. On top of it, it needs experts for its maintenance.

KEYWORDS

1. **Expert system:** It is an AI-based computer program that simulates expert judgment by the use of knowledge and inference mechanism to offer an advice or infer.
2. **Inference mechanism:** It is responsible for getting conclusions with the help of KB.
3. **Domain exploration:** It is concerned with the exploration of the knowledge about the domain in which the expert system is active.
4. **Expertise transfer:** In the process of expert system building, the expertise is transferred from one entity to another in the form of knowledge to build KB.
5. **Meta-knowledge:** In case of expert systems, the meta-knowledge helps in building a domain independent system.
6. **MYCIN:** It is an early expert system that was developed to aid the physicians in diagnosis and treatment of meningitis and bacterial infections.
7. **DENDRAL:** It is also an early expert system that analyses the organic compounds so as to determine their structure.
8. **Knowledge acquisition tools:** These tools assist and help in getting the knowledge from the expert.
9. **Shells:** These are the interpreters used for building an expert system.
10. **Explanations:** These are the features of an expert system that tell about the reasoning.
11. **Self-explaining system:** It is the ability of an expert system to justify the rules it has fired.
12. **Rule-based expert system:** It is an expert system that is based on if-then rules, facts and the inference mechanism.
13. **Frame-based expert system:** It is an expert system that makes use of the knowledge as frames.
14. **Fuzzy expert system:** It is an expert system that makes use of fuzzy logic consisting of collection of membership functions and rules to reason.

MULTIPLE CHOICE QUESTIONS

(*Note:* There can be more than one answer.)

1. MYCIN falls under the category of
   (a) Shell
   (b) Rule-based expert system
   (c) Frame-based expert system
   (d) None of these
2. A rule of 'If you are wearing a cardigan, then it is cold' falls under the semantics of
   (a) Recommendation
   (b) Heuristic
   (c) Relation
   (d) Directive
3. The core part of decision-making for the expert system lies in the
   (a) Knowledge base
   (b) Explanations
   (c) Inference mechanism