IDHAYA COLLEGE FOR WOMEN
KUMBAKONAM - 612 001

PG & RESEARCH DEPARTMENT OF COMPUTER SCIENCE

ACADEMIC YEAR : **2019-2020**

SEMESTER : **II**

CLASS : **I M.Sc (CS)**

SUBJECT INCHARGE : **Dr. C.PREMILA ROSY**

SUBJECT NAME : **ARTIFICIAL INTELLIGENCE**

SUBJECT CODE : **P16CSE2B**

UNIT V

Game playing – The minimax search procedure – Expert System - Perception and Action

UNIT -V

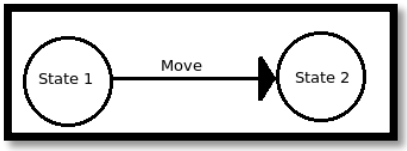
GAME PLAYING

GAME PLAYING

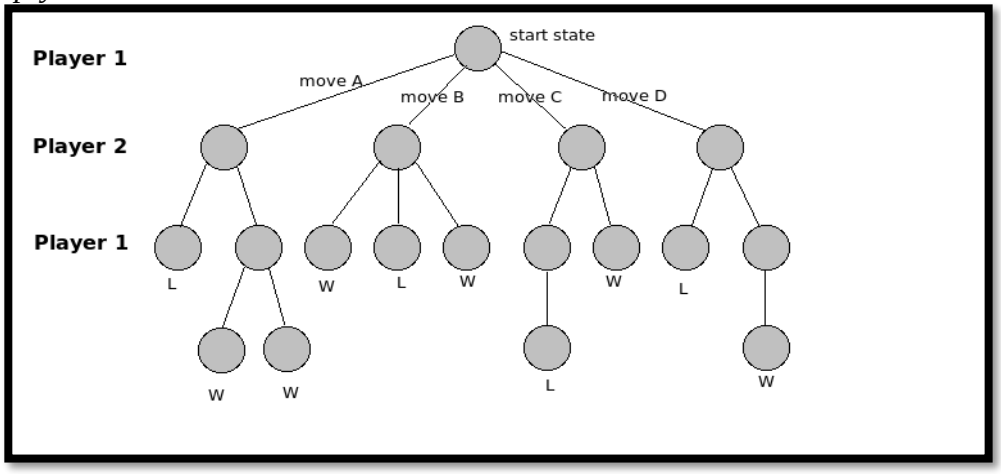
Game playing is a search problem Defined by

- Initial state
- Successor function
- Goal test
- Path cost / utility / payoff function

- For most cases the most convenient way to represent game play is on a graph.
- We will use graphs with nodes representing game "states" (game position, score, etc.) and edges representing a move by a player that moves the game from one state to another:



- Using these conventions, we can turn the problem of solving a game into a version of graph search, although this problem differs from other types of graph search.
- For instance, in many cases we want to find a single state in a graph, and the path from our start state to that state, whereas in game search we are not looking for a single path, but a winning move.
- The path we take might change, since we cannot control what our opponent does.
- Below is a small example of a game graph.
- The game starts in some initial state at the root of the game tree.
- To get to the next level, player one chooses a move, A, B, C, or D.
- To get to the next level, player two makes a move, etc. Each level of the tree is called a *ply*.



So, if we are player one, our goal is to find what move to take to try to ensure we reach one of the "W" states. Note that we cannot just learn a strategy and specify it beforehand, because our opponent can do whatever it wants and mess up our plan.

When we talk about game graphs some terms you might want to be familiar with are:

- Branching factor (b)

  The number of outgoing edges from a single node. In a game graph, this corresponds to the number of possible moves a player can make. So, for instance, if we were graphing tic-tac-toe, the branching factor would be 9 (or less, since after a person moves the possible moves are limited, but you get the idea)

- Ply

  A level of the game tree. When a player makes a move the game tree moves to the next ply.

- Depth (d)

  How many ply's we need to go down the game tree, or how many moves the game takes to complete. In tic-tac-toe this is probably somewhere around 6 or 7 (just made that up...). In chess this is around 40.

Solving a Game

We often talk about the notion of "solving" a game. There are three basic types of "solutions" to games:

1. Ultra-weak- The result of perfect play by each side is known, but the strategy is not known specifically.
2. Weak- The result of perfect play and strategy from the start of the game are both known.
3. Strong -The result and strategy are computed for all possible positions.

THE MINMAX SEARCH PROCEDURE

The most used game tree search is the *minimax* algorithm. To get a sense for how this works, consider the following:

Helen and Stavros are playing a game. The rules of this game are very mysterious, but we know that each state involves Helen having a certain number of drachmas at each state. Poor Stavros never gets any drachmas, but he doesn't want Helen to get any richer and keep bossing him around.

So, Helen wants to maximize her drachmas, while Stavros wants to minimize them. What should each player do? At each level Helen will choose the move leading to the greatest value, and Stavros will move to the minimum-valued state, hence the name "minimax."

Formally, the minimax algorithm is described by the following pseudocode:

```
def max-value(state,depth):
    if (depth == 0): return value(state) v = -infinite
    for each s in SUCCESSORS(state):
    v = MAX(v,min-value(s,depth-1))
    return v


def min-value(state,depth):
    if (depth == 0): return value(state) v = infinite
    for each s in SUCCESSORS(state): v = MIN(v,max-
       value(s,depth-1))
    return v
```

EVALUATION FUNCTIONS

- Evaluation functions, besides the problem above of finding the optimal ordering of game states to explore, is perhaps the part of game search/play that involves the most actual thought (as opposed to brute force search).
- These functions, given a state of the game, will compute a value based only on the current state, and cares nothing about future or past states.

As an example, evaluation, consider one of the type that Shannon used in his original work on solving chess. His function (from White's perspective) calculates the value for white as:

- +1 for each pawn
- +3 for each knight or bishop
- +5 for each rook
- +9 for each queen
- + some more points based on pawn structure, board space, threats, etc.

For many games the evaluation of certain game positions has been stored in a huge database that is used to try to "solve" the game. A couple examples are:

- OHex - partial solutions to Hex games
- Chinook - database of checkers positions

EXPERT SYSTEMS

The expert systems are the computer applications developed to solve complex problemsin a particular domain, at the level of extra-ordinary human intelligence and expertise.

Characteristics of Expert Systems

- High Performance
- Understandable
- Reliable
- Highly responsive

Capabilities of Expert Systems

- ✓ The expert systems are capable of – Advising

- ✓ Instructing and assisting human in decision making Demonstrating
- ✓ Deriving a solution Diagnosing Explaining Interpreting input Predicting results
- ✓ Justifying the conclusion
- ✓ Suggesting alternative options to a problem

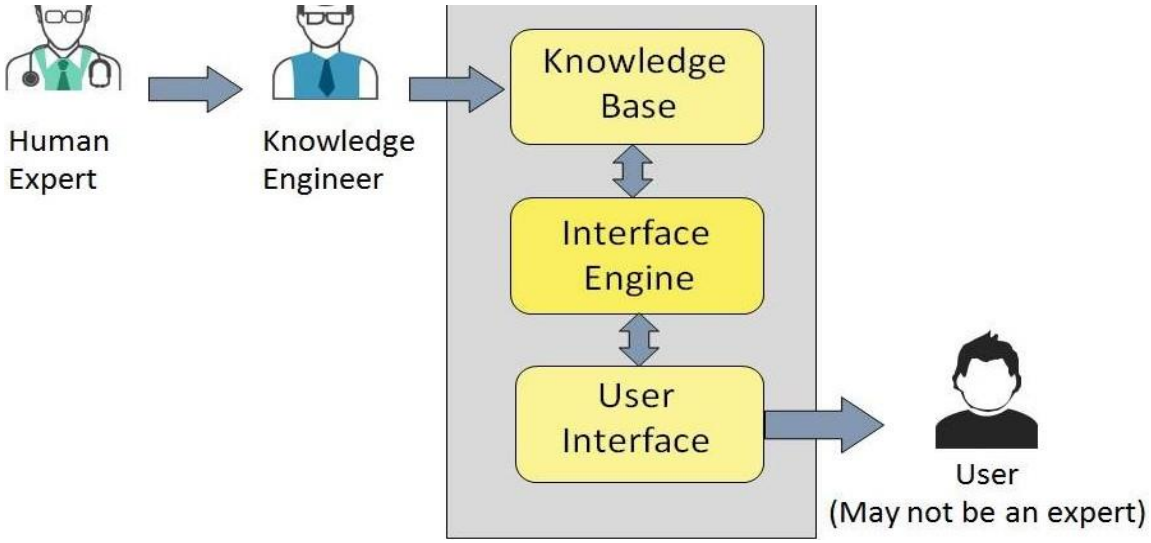
- ✓ They are incapable of Substituting human decision makers Possessing human capabilities
- ✓ Producing accurate output for inadequate knowledge base

  Refining their own knowledge

Components of Expert Systems

The components of ES include

- ✓ Knowledge Base
- ✓ Interface Engine User Interface



Knowledge Base

- ➢ It contains domain-specific and high-quality knowledge.
- ➢ Knowledge is required to exhibit intelligence.
- ➢ The success of any ES majorly depends upon the collection of highly accurate and precise knowledge.

What is Knowledge?

- ➢ The data is collection of facts.

- The information is organized as data and facts about the task domain.
- **Data, information,** and **past experience** combined together are termed as knowledge.

Components of Knowledge Base

The knowledge base of an ES is a store of both, factual and heuristic knowledge.

- **Factual Knowledge**− It is the information widely accepted by the Knowledge Engineers and scholars in the task domain.
- **Heuristic Knowledge** − It is about practice, accurate judgement, one's ability of evaluation, and guessing.

Knowledge representation

- It is the method used to organize and formalize the knowledge in the knowledge base. It is in the form of IT-THEN-ELSE rules.

Knowledge Acquisition

- The success of any expert system majorly depends on the quality, completeness, and accuracy of the information stored in the knowledgebase.
- The knowledge base is formed by readings from various experts, scholars, and the **Knowledge Engineers**.
- The knowledge engineer is a person with the qualities of empathy, quick learning, and case analyzing skills.

He acquires information from subject expert by recording, interviewing, and observing him at work, etc. He then categorizes and organizes the information in a meaningful way, in the form of IF-THEN-ELSE rules, to be used by interference machine. The knowledge engineer also monitors the development of the ES.

Interface Engine

- Use of efficient procedures and rules by the Interface Engine is essential in deducting a correct, flawless solution.
- In case of knowledge-based ES, the Interface Engine acquires and manipulates the knowledge from the knowledge base to arrive at a particular solution.

Applies rules repeatedly to the facts, which are obtained from earlier rule application. Adds new knowledge into the knowledge base if required.

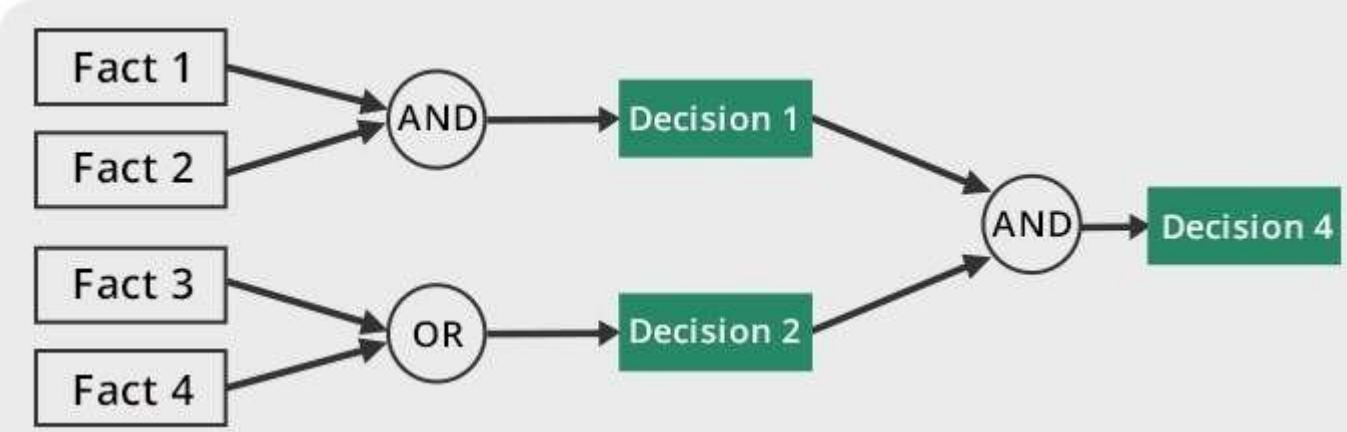
Resolves rules conflictwhen multiple rules are applicable to a particular case.

To recommend a solution, the interface engine uses the following strategies

1. Forward Chaining
2. Backward Chaining

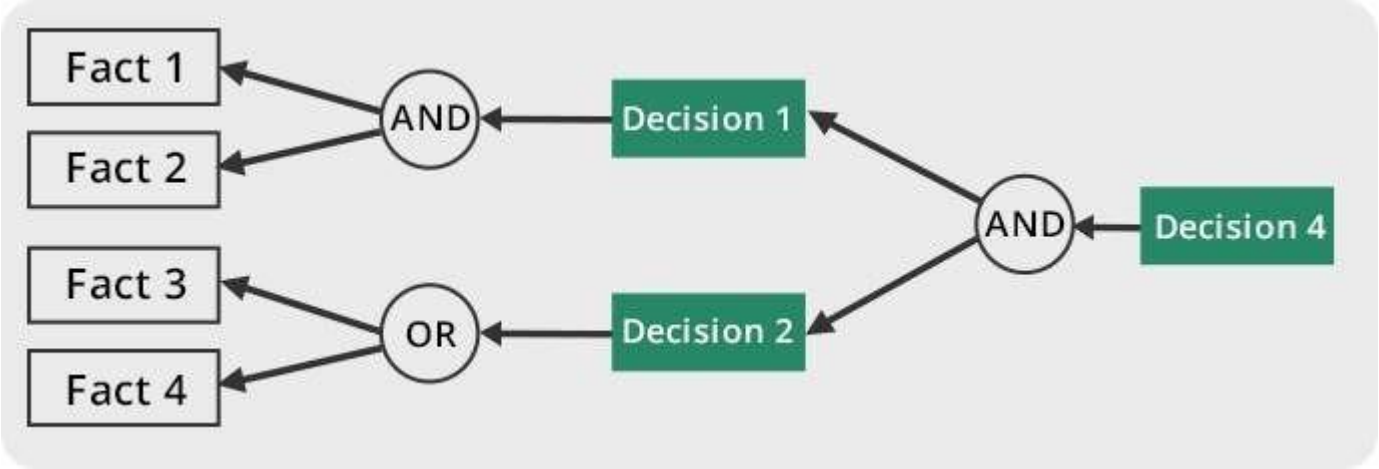
Forward Chaining

- It is a strategy of an expert system to answer the question, **"What can happen next?"**
- Here, the interface engine follows the chain of conditions and derivations and finally deduces the outcome. It considers all the facts and rules and sorts them before concluding to a solution.
- This strategy is followed for working on conclusion, result, or effect. For example, prediction of share market status as an effect of changes in interest rates.



Backward Chaining

- With this strategy, an expert system finds out the answer to the question, **"Why this happened?"**
- On the basis of what has already happened, the interface engine tries to find out which conditions could have happened in the past for this result.
- This strategy is followed for finding out cause or reason.
- For example, diagnosis of blood cancer in humans.



User Interface

- User interface provides interaction between user of the ES and the ES itself. It is generally

- Natural Language Processing so as to be used by the user who is well-versed in the task domain. The user of the ES need not be necessarily an expert in Artificial Intelligence.
- It explains how the ES has arrived at a particular recommendation. The explanation may appear in the following forms −

Expert Systems Limitations

- No technology can offer easy and complete solution. Large systems are costly, require significant development time, and computer resources.
- Limitations of the technology Difficult knowledge acquisition ES are difficult to maintain High development costs
- Applications of Expert System
- The following table shows where ES can be applied.

Application		Description
Design Domain	-	Camera lens design, automobile design.
Medical Domain	-	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Monitoring Systems	-	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Finance/Commerce	-	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

PERCEPTION AND VISION

Algorithm	Focus scheme that implements it
Case-based reasoning	Memory-based simulations
Prediction	Forward simulation
Counterfactual reasoning	Counterfactual simulation
Backtracking search	(nested) Counterfactual simulation.
Backward chaining/Theorem Proving	Antecedent simulation
Truth maintenance	Resimulation
Bayesian inference	Stochastic simulation

Stimulus	Representation/ Algorithm	Action
Sensors	[None for reactive systems]	Action
Cause	Causal Rules / Rule Matching	Assert effect
Input layers	Neural Networks / Network propagation	Output layer
Object category information	Category hierarchies/ Graph walking	Assert other categories the object belongs to
Two temporal intervals	Temporal intervals / Constraint propagation	Assert the possible constraints between the two Intervals

**********ALL THE BEST**********