ST.FRANCIS INSTITUTE OF TECHNOLOGY
Mount Poinsur, S.V.P. Road, Borivli (West), Mumbai - 400103
Computer Engineering Department
Academic Year: 2021-2022 **Class/Branch**: BE CMPN **Subject:** CSC 703 Artificial Intelligence & Soft Computing **Semester**: VII

Name: Selas Moro **PID:** 182074 **Roll:**39 **Class:** BECMPNB

Experiment No. 8

Case Study on Expert Systems

Aim: To study and understand the working of expert systems.

Theory:

Expert Systems are computer programs built for commercial application using the programming techniques of artificial intelligence.

Examples: There are many examples of expert system. Some of them are given below:
● MYCIN: One of the earliest expert systems based on backward chaining. It can identify various bacteria that can cause severe infections and can also recommend drugs based on the person's weight.
● DENDRAL: It was an artificial intelligence based expert system used for chemical analysis. It used a substance's spectrographic data to predict it's molecular structure.
● R1/XCON: It could select specific software to generate a computer system wished by the user.

Features of Expert systems

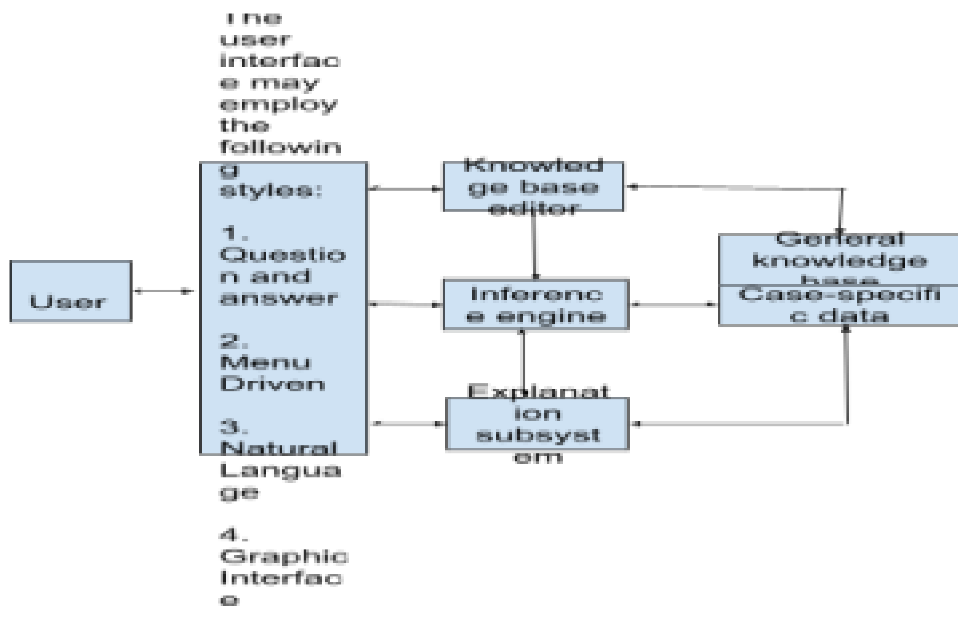
- Facility of solving problems that require an expertise
- Speedy Solutions
- Reliable Solutions
- Cost Reduction
- Elimination of uncomfortable and monotonous operations
- Power to manage without human experts
- Wider access to knowledge.

When to use Expert Systems?

- Human experts are difficult to find
- Human experts are expensive
- Knowledge improvement is needed
- Knowledge is difficult to acquire
- Poor available information
- Problem is subjected to change

Basic Elements of Expert System

An expert system is typically composed of at least three primary components. These are the inference engine, the knowledge base, and the user interface.



1. Knowledge Base : The knowledge base is a collection of rules or other information structures derived from the human expert. Rules are typically structured as If/Then statements of the form:

IF <antecedent> THEN <consequent>

The antecedent is the condition that must be satisfied. When the antecedent is satisfied, the rule is triggered and is said to "fire". The consequent is the action that is performed when the rule fires.

2. Agenda : When rules are satisfied by the program, they are added to a queue called the agenda. The agenda is an unordered list of all the rules whose antecedents are currently satisfied. Knowledge bases are typically not ordered, because order tends to play very little role in an expert system. Rules may be placed on the agenda in any order, and they may be fired in any order once they are on the agenda.

3. Inference Engine : The inference engine is the main processing element of the expert system. The inference engine chooses rules from the agenda to fire. If there are no rules on the agenda, the inference engine must obtain information from the user in order to add more rules to the agenda. It makes use of knowledge base, in order to draw conclusions for situations. It is responsible for gathering the information from the user, by asking various questions and applying it wherever necessary. It seeks information and relationships from the knowledge base and to provide answers, predictions and suggestions the way a human expert would.

4. User Interface : A user interface is the method by which the expert system interacts with a user. These can be through dialog boxes, command prompts, forms, or other input methods.

5. Working Memory (Database/ Case Specific Data)

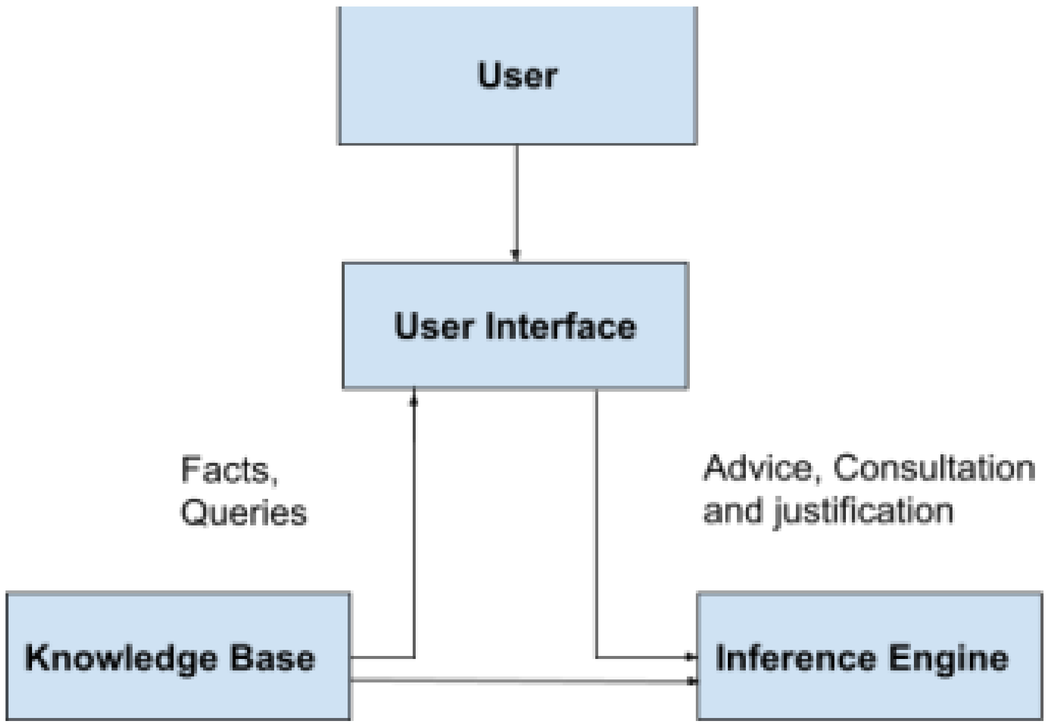
Working memory contains the data that is received from the user during the expert system session. Values in working memory are used to evaluate antecedents in the knowledge base. Consequences from rules in the knowledge base may create new values in working memory, update old values, or remove existing values.

6. Explanation Mechanism

The method by which an expert system reaches a conclusion may not be obvious to a human user, so many expert systems will include a method for explaining the reasoning process that leads to the final answer of the system.

Architecture of an Expert System

The architecture of expert system reflects the knowledge engineers way of knowledge representation and performance of intelligent decision making.



This architecture is independent of computer hardware. The user interface allows the system users to

i. Enter rules and facts about a particular situation
ii. Ask questions of the system
iii. Provide responses to the user requests
iv. Support all other communication between the system and the user

The knowledge is contained in codified form within the KB which can be easily read and understood. The inference engine uses the information provided to it by the KB and the user to infer new facts.

This architecture can be expanded by extending KB into a knowledge database and domain database. The knowledge base contains rules and domain database contain facts. The KB is updated so that the system provides the most relevant and complete assistance to the user. Some expert system dedicate a module to update the KB known as Knowledge Acquisition Facility. It provides a dialogue with human experts to acquire knowledge in terms of facts and rules and place them in the domain database and the knowledge database respectively.

Self Training Facility: This facility accepts the facts developed by the inference engine and compares derived facts with that in the domain database and if the knowledge is new, it is upgraded.

Stages in the development of an Expert System

The different stages in the development of an expert system are described as below:

1. **Outline Statement:** This stage identifies an appropriate system. The expert and knowledge engineer work out the concepts, boundaries, relationships and control mechanisms to be included in the system.Development strategies, constraints and user expectations are explored. The results can be documented in an outline specification for initial prototype development
2. **Knowledge Acquisition :** The experts and knowledge engineers interact intensively in this stage. The experts highlight the essential issues and knowledge engineer tries to comprehend the essence of the knowledge, its limits and its complexities.
3. **Knowledge Representation:** Once the knowledge engineer has enough knowledge about the system, he has to design a method for appropriate knowledge representation. It is knowledge engineer's ability to enter into the frame of reference of both expert and user and to suitably structure the acquired knowledge.
4. **Prototype Development :** Most systems develop a prototype model. The advantage of developing prototype is that we will know whether the system is feasible or not. The users get an opportunity to test the system and whether it is likely to meet their requirements. The developers also gets an opportunity to evaluate the cost and performance of the chosen system.
5. **Testing:** Testing involves evaluating the performance and utility of the prototype program and revising it as necessary. The prototype should be tested on many problems to evaluate its performance and utility. It may uncover problems such as missing concepts and relations in the representation scheme, knowledge represented at the wrong level of detail.,or infeasible control mechanism. The developers may recycle through different development phases by reformulating the concepts, refining the inference rules and retesting the control flow.
6. **Main Knowledge Acquisition:** Once the prototype is reviewed and tested, the actual system is developed. This stage assess the extent of the knowledge that is required to meet users needs.

7. **Specification with Detailed Information:** The detailed specification covers the objectives of the expanded system, the resources required, the projected time required for implementation, planned costs, system testing and implementation planning.
8. **System Development:** During this stage, it is very important for the users to know exactly how the system is progressing, any problems encountered, and the evidence of new limitations and opportunities. This stage requires careful monitoring.
9. **Implementation:** Implementation procedures should be carried out by the user and supported by the expert. The implementation plan should have been documented during the specification stage.
10. **Maintenance:** It requires continuous revision and updating to ensure that the knowledge it contains is always up to date and in accordance with the changing environment in which the organization operates.

Experiment Exercise : Identify an expert system and write note on the User Interface, Inference Engine and Knowledge Base of the System

Topic: ALFHA GO

AlphaGo

Report:

Need of the system:

A long-standing goal of artificial intelligence is an algorithm that learns, tabula rasa, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance, or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher quality move selection and stronger self-play in the next iteration. Starting tabula rasa, our new program AlphaGo Zero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

Knowledge Base:

As of 2016, AlphaGo's algorithm uses a combination of machine learning and tree search techniques, combined with extensive training, both from human and computer play. It uses Monte Carlo tree search, guided by a "value network" and a "policy network," both implemented using deep neural network technology. A limited amount of game-specific feature detection pre-processing (for example, to highlight whether a move matches a nakade pattern) is applied to the input before it is sent to the neural networks

The system's neural networks were initially bootstrapped from human gameplay expertise. AlphaGo was initially trained to mimic human play by attempting to match the moves of expert players from recorded historical games, using a database of around 30 million moves. Once it had reached a certain degree of proficiency, it was trained further by being set to play large numbers of games against other instances of itself, using reinforcement learning to improve its play. To avoid "disrespectfully" wasting its opponent's time, the program is specifically programmed to resign if its assessment of win probability falls beneath a certain threshold; for the match against Lee, the resignation threshold was set to 20%.

Then in AlphaGo
Rather than training on an existing dataset, AlphaGo Zero uses a novel form of Reinforcement Learning wherein the system is its own teacher. The system starts off with a neural network that knows nothing about the game of Go. It then plays games against itself, by combining this neural network with a powerful search algorithm. As it plays, the neural network is tuned and updated to predict moves, as well as the eventual winner of the games. This updated neural network is then recombined with the search algorithm to create a new, stronger version of AlphaGo Zero, and the process begins again. In each iteration, the performance of the system improves by a small amount, and the quality of the self-play games increases, leading to more and more accurate neural networks and ever stronger versions of AlphaGo Zero. This technique is more powerful than previous versions of AlphaGo because it is no longer constrained by the limits of human knowledge. Instead, it is able to learn tabula rasa from the strongest player in the world: AlphaGo itself.

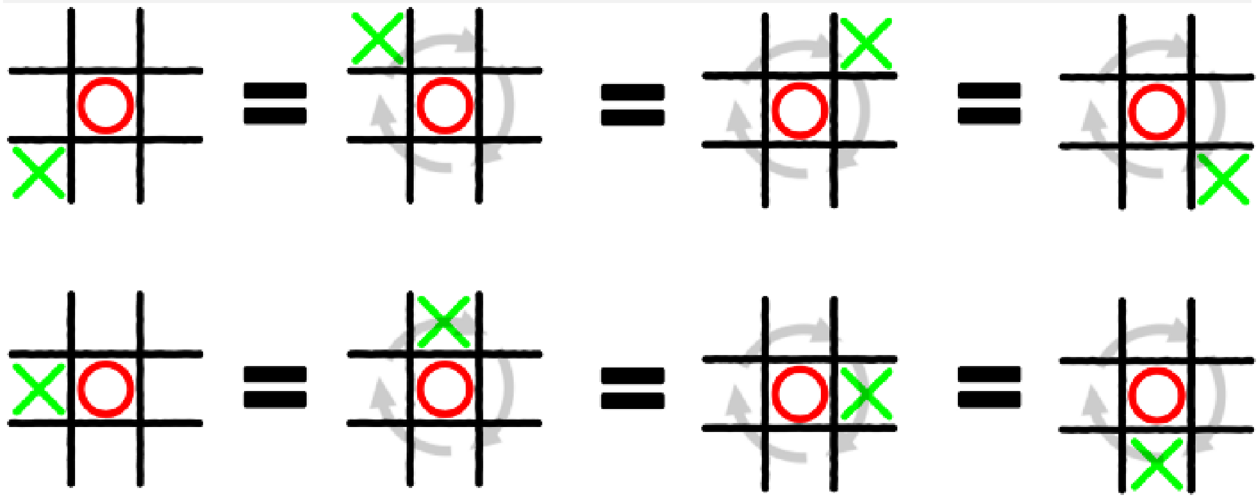
Inference Engine:

The state machine, usually referring to a Finite State Machine, is an often-underestimated concept in-game AI nowadays especially after machine learning gained huge popularity over the years. However, such old-school design dominated with a win rate of 96.15% in the SSCAIT 2018 challenge playing against other StarCraft AI using machine learning.

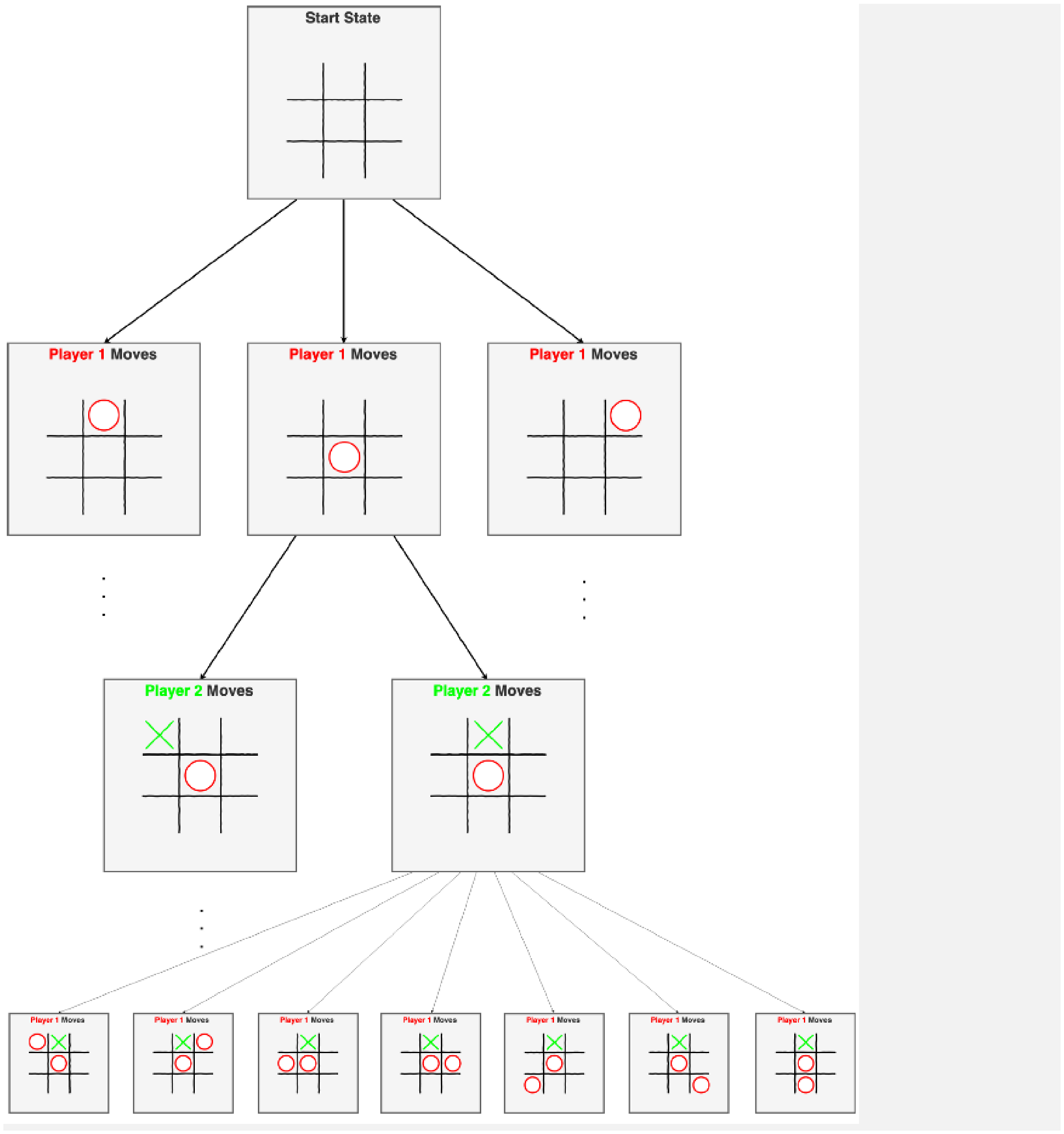
Outside the field of game AI, state machines are also applied widely into fields such as this cool phone assistant AI from Google. In this section, we will go into the theory behind all such wonders.

In the field of computer science, we also value the efficiency of the algorithms. Less amount of computation and memory consumption means a lighter load on the hardware and electricity consumption, resulting in a reduced cost running the algorithm. And thus, rather than going over every possible state, there are many ways to simplify the problem.

For example, the 8 states of tic-tac-toe shown below are rotationally equivalent, and thus we can consider them as 2 states instead. These types of translational and rotational equivalencies will be discussed in more detail as we dive into convolutional neural networks in a later section.

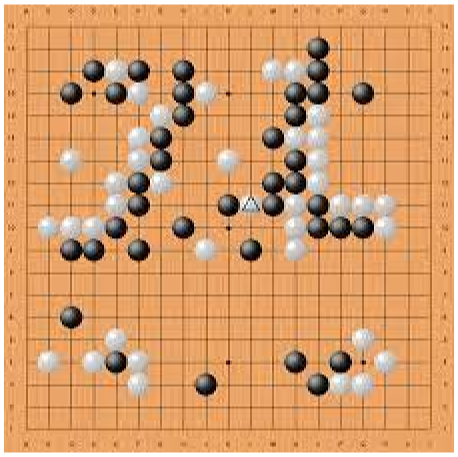


By eliminating the extra states, the size of the search tree can be greatly reduced. There are over 3500 states in the first 4 layers of the search tree if such a reduction is not applied. Searching through 3500 states is not much of a task for a modern computer, but the number grows exponentially and can go to the scale of microseconds of the age of our universe when it goes to a bigger board such as chess.



User Interface:

The user interface of AlphaGo is a Go game board where the user selects the opponent’s move and the system generates its move.



This powerful technique is no longer constrained by the limits of human knowledge. Instead, the computer program accumulated thousands of years of human knowledge during a period of just a few days and learned to play Go from the strongest player in the world, AlphaGo.

AlphaGo Zero quickly surpassed the performance of all previous versions and also discovered new knowledge, developing unconventional strategies and creative new moves, including those which beat the World Go Champions Lee Sedol and Ke Jie. These creative moments give us confidence that AI can be used as a positive multiplier for human ingenuity.

Two players, using either white or black stones, take turns placing their stones on a board. The goal is to surround and capture their opponent's stones or strategically create spaces of territory. Once all possible moves have been played, both the stones on the board and the empty points are tallied. The highest number wins.

As simple as the rules may seem, Go is profoundly complex. There are an astonishing 10 to the power of 170 possible board configurations - more than the number of atoms in the known universe. This makes the game of Go a googol times more complex than chess.

Post Experiment Exercise : Find out the programming language in which the expert system was designed and built.

Programming Language used:
As of 2016, AlphaGo's algorithm uses a combination of machine learning and tree search techniques, combined with extensive training, both from human and computer play. It uses Monte Carlo tree search, guided by a "value network" and a "policy network," both implemented using deep neural network technology.
The major programming languages used are: Python, C++.

Conclusion: The students will be able to understand the working of expert system and under the context of the above expert system (i.e. ALPHAGO) , We were able to get vivid knowledge about the expert system and their Fundamentals.