INT404 ARTIFICIAL INTELLIGENCE

Lecture 5

General Problem Solver

- General Problem Solver (GPS) was an AI program proposed by Herbert Simon, J. C. Shaw, and Allen Newell.

- The goal was to make it work as a universal problem-solving machine.

- GPS was supposed to solve all the problems using the same base algorithm for every problem.

- To program the GPS, authors created a new language called Information Processing Language.

General Problem Solver

- The basic idea was to express any problem with a set of well-formed formulas.
- Well-formed formulas would be a part of a directed graph with multiple sources (start) and sinks (end).
- In GPS, source refers to axioms and sink refers to conclusions.
- GPS can solve only well-defined problems, such as theorem proving etc.
- GPS became intractable because in real world in solution of the problem number of possible path can be taken.
- If it tries brute force it become computationally infeasible.
- For example, let us solve the problem to get some milk from the grocery store.

General Problem Solver

Let's see how to structure a given problem to solve it using GPS:

1. The first step is to define the goals. Let's say our goal is to get some milk from the grocery store.
2. The next step is to define the preconditions. These preconditions are in reference to the goals. To get milk from the grocery store, we need to have a mode of transportation and the grocery store should have milk available.
3. After this, we need to define the operators. If my mode of transportation is a car and if the car is low on fuel, then we need to ensure that we can pay the fueling station. We need to ensure that you can pay for the milk at the store.

An operator takes care of the conditions and everything that affects them. It consists of actions, preconditions, and the changes resulting from taking actions.

Building Goal-Based Agents

- We have a **goal** to reach
  - Driving from point A to point B
  - Put 8 queens on a chess board such that no one attacks another
  - Prove that John is an ancestor of Mary
- We have information about where we are now at the **beginning.**
- We have a set of **actions/operators** we can take to move around (change from where we are)
- **Objective**: find a sequence of legal actions/operators which will bring us from the start point to a goal

Operator(What are the actions?)

- Quantify (Express as a number, measure or quantity) all of the primitive actions or events that are sufficient to describe all necessary changes in solving a task/goal.
- No uncertainty associated with what an action does to the world. That is, given an action (**operator or move**) and a description of the current state of the world, the action completely specifies
  - **Precondition:** if that action CAN be applied to the current world (i.e., is it applicable and legal), and
  - **Effect:** what the exact state of the world will be after the action is performed in the current world (i.e., no need for "history" information to be able to compute what the new world looks like).

Actions

- Note also that actions can all be considered as **discrete events** that can be thought of as occurring at an **instant of time**.
  - That is, the world is in one situation, then an action occurs and the world is now in a new situation.
  - For example, if "Mary is in class" and then performs the action "go home," then in the next situation she is "at home." There is no representation of a point in time where she is neither in class nor at home (i.e., in the state of "going home").
- The number of operators needed depends on the **representation** used in describing a state.

Representing states

- At any moment, the relevant world is represented as a **state**
  - Initial (start) state: **S**
  - An action (or an **operation**) changes the current state to another state (if it is applied): state transition
  - An action can be taken (applicable) only if the its precondition is met by the current state
  - For a given state, there might be more than one applicable actions
  - **Goal state**: a state satisfies the goal description or passes the goal test
  - **Dead-end state**: a non-goal state to which no action is applicable

Representing states

- **State space**:
  - Includes the initial state S and all other states that are reachable from S by a sequence of actions
  - A state space can be organized as a graph:

    nodes: states in the space

    arcs: actions/operations
- The **size of a problem** is usually described in terms of the **number of states** (or the size of the state space) that are possible.
  - Tic-Tac-Toe has about 3^9 states.
  - Checkers has about 10^40 states.
  - Rubik's Cube has about 10^19 states.
  - Chess has about 10^120 states in a typical game.

Formalizing Search in a State Space

- A state space is a **graph**, (V, E) where V is a set of **nodes** and E is a set of **arcs**, where each arc is directed from a node to another node
- **node:** corresponds to a **state**
  - –state description
- **arc:** corresponds to an applicable action/operation.
  - –the source and destination nodes are called as **parent (immediate predecessor)** and **child** (**immediate successor**) nodes with respect to each other
  - –each arc has a fixed, non-negative **cost** associated with it, corresponding to the cost of the action

Definition 1 [Generation of a node] A node nn is generated when a data structure representing its contents is created, i.e., when it is allocated in main memory.

Definition 2 [Expansion of a node] A node nn is expanded when all its children are generated.

- One or more nodes are designated as **start nodes**
- A **goal test** predicate is applied to a node to determine if its associated state is a goal state
- A **solution** is a sequence of operations that is associated with a path in a state space from a start node to a goal node
- The **cost of a solution** is the sum of the arc costs on the solution path

For Example:

If one wants to make a cup of coffee then what he will do?

State space representation of this problem can be done as follow:

First of all analyze the problem i.e. verify whether the necessary ingredients like instant coffe powder, milk powder, sugar, kettle, stove etc are available or not.

If are available, them steps to solve the problems are.

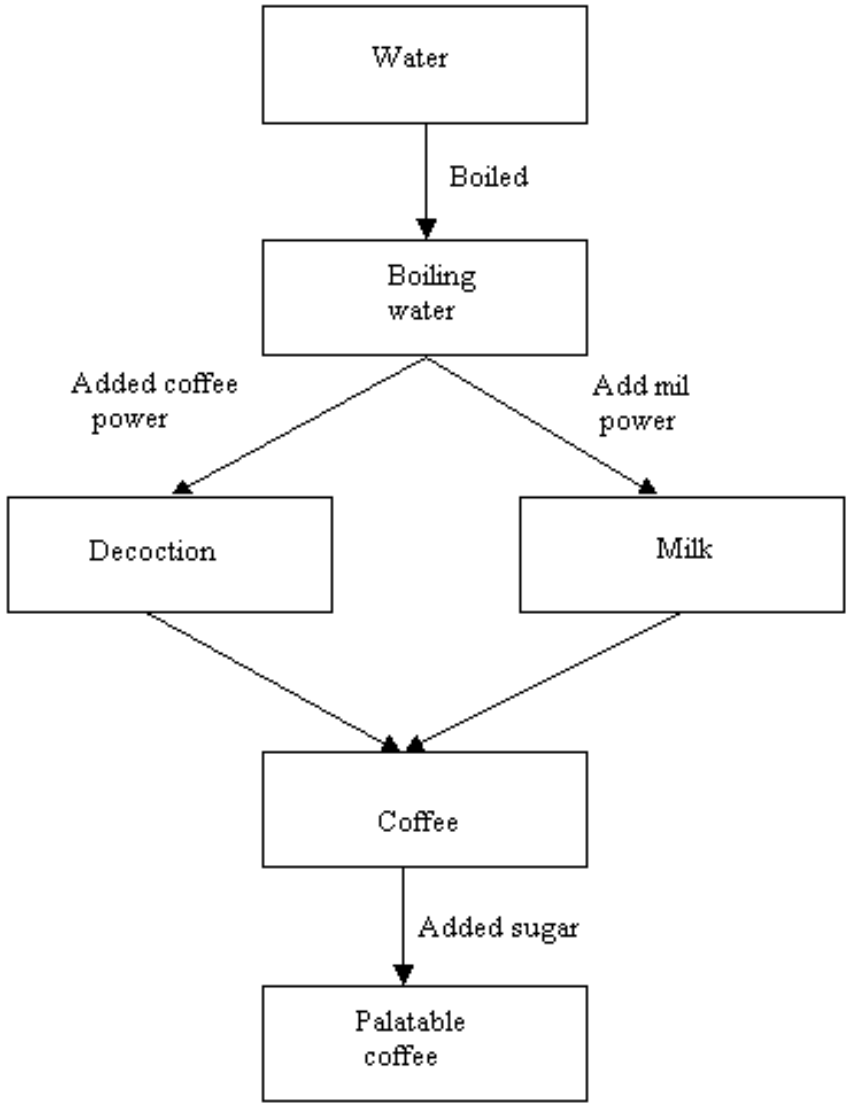
Boil necessary water in the kettle.

Take same of the boiled water in a cup add necessary amount of instant coffee pour to decoction(*the extraction of water-soluble drug substances by boiling).*

Add milk pour to the remaining boiling water to make milk.

Add sufficient quantity of sugar to your taste & coffee is ready.

SIMPLE EXAMPLE FOR THE STATE SPACE



Define knowledge:

Formally speaking, a piece of knowledge is a **function** that maps a **domain of clauses** onto a **range of clauses.**

The function may be **algebraic or relational** that depends upon the area of applications.

For example consider the rule PR1, which maps mother child relationship between the same pair.

i.e. PR1: Mother (m, c) → Loves (m, c)

Where in PR1 Mother (m, c) and loves (m, c) are two predicates; ‘m’, ‘c’ are variables and ‘→’ is an if-then operator.

When m and c assume specific values for example m = Sita and c = Lob then **Mother (Sita, Ravi) and loves (Sita, Ravi) are called clauses** (i.e. simple sentence/simple provision in law.).

Production system:

❖A production system provides storing and searching of knowledge.

❖ Production system is the simplest and one of the oldest techniques of knowledge representation.

Structure of PS:

1. A set of production rules (PR):

Left side determines the applicability of the rule and a right side describes the operation to be performed if the rule is applied.

2. One or more knowledge/database:

That contain whatever information is appropriate for the particular task.

3. **A control structure/strategies/interpreter:**

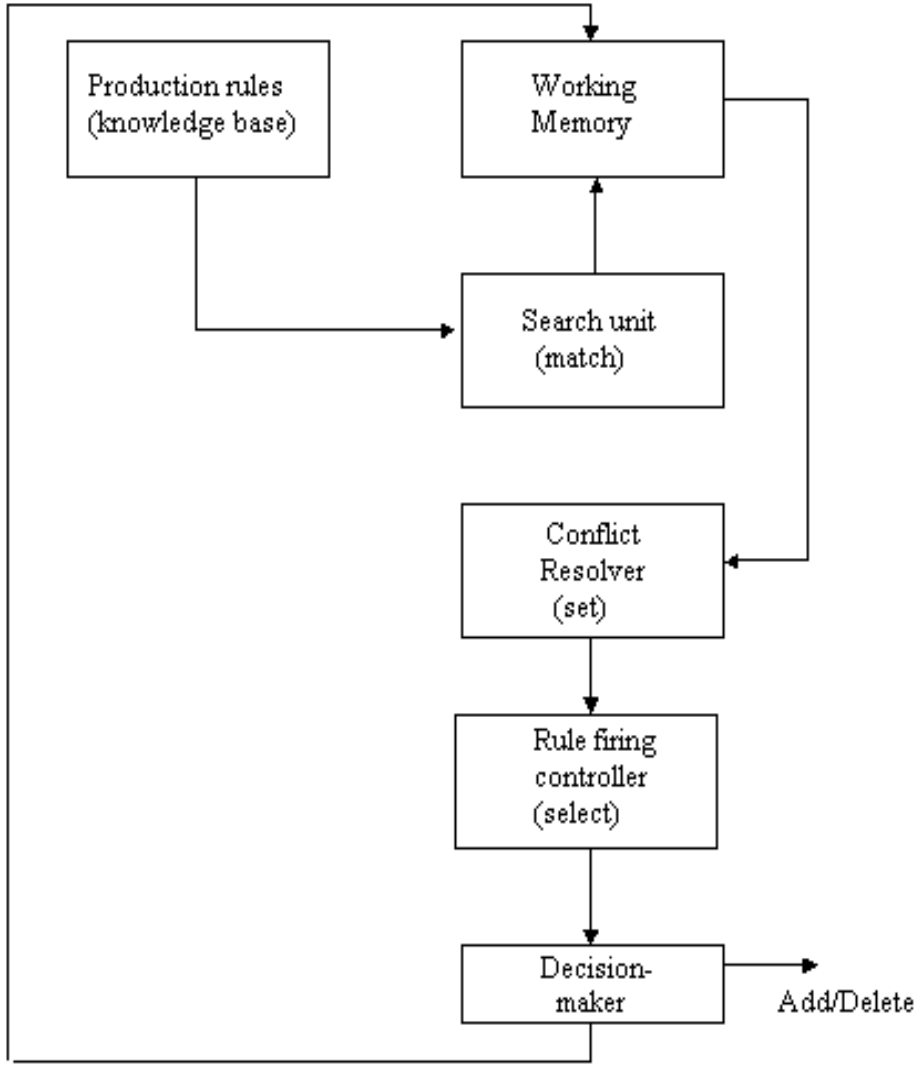
The control structure is strategy that specifies the order in which the rules will be compared to the database and also a way of resolving the conflicts that arise when several rules match at once.

- *the first requirement of a good control strategy is that it causes motion.*
- *The second requirement of a good control strategy is that it be systematic.*

4. **Rule applier:**

A conflict may arise when more than one rule that can be fired in a situation. Rule interpreter is to decide which is to be served of what is the order. The strategies used to resolve the conflict resolution strategies.

Architecture of a production system:



❖**Perform the first system:** To choose first rule that matches.

❖**Sequencing technique:** To adopt the rule in the sequence they are.

❖**Matching rules:** If there are two matching rules one is more specific than the other, activate it.

❖**More recent policy:** It is generally believed that a newly added rule is more knowledgeable than existing ones. Hence is system is adopting this method it should be for most recent rules.

It has wide application in automata theory, formal grammars and the design of programming languages. However, it has been entered into knowledge engineering by Buchanan and Feigenvan.