STATE SPACE REPRESENTATION

INT404

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** we can take to move around (change from where we are)
- **Objective**: find a sequence of legal actions which will bring us from the start point to a goal

What are the actions?

- Quantify 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

•**node generation:** making explicit a node by applying an action to another node which has been made explicit

•**node expansion:** generate **all** children of an explicit node by applying **all** applicable operations to that node

•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

State Space

- In state space representation of problem start with the initial state & wandering through the set of states in search of final state.
- The state space representation forms the basis of most of AI problems. It's structure corresponds to structure of problem solving in two importance ways.
- It allows for formal destination of a problem as the need to convert some given situation into some desired situation using a set of permissible operation.
- It permit us to a design the process of solving a particular problem as a combination

For Example:

It 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 cases power milk power, sugar, kettle stove etc are available or not.

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

Boil necessary water in the kettle.

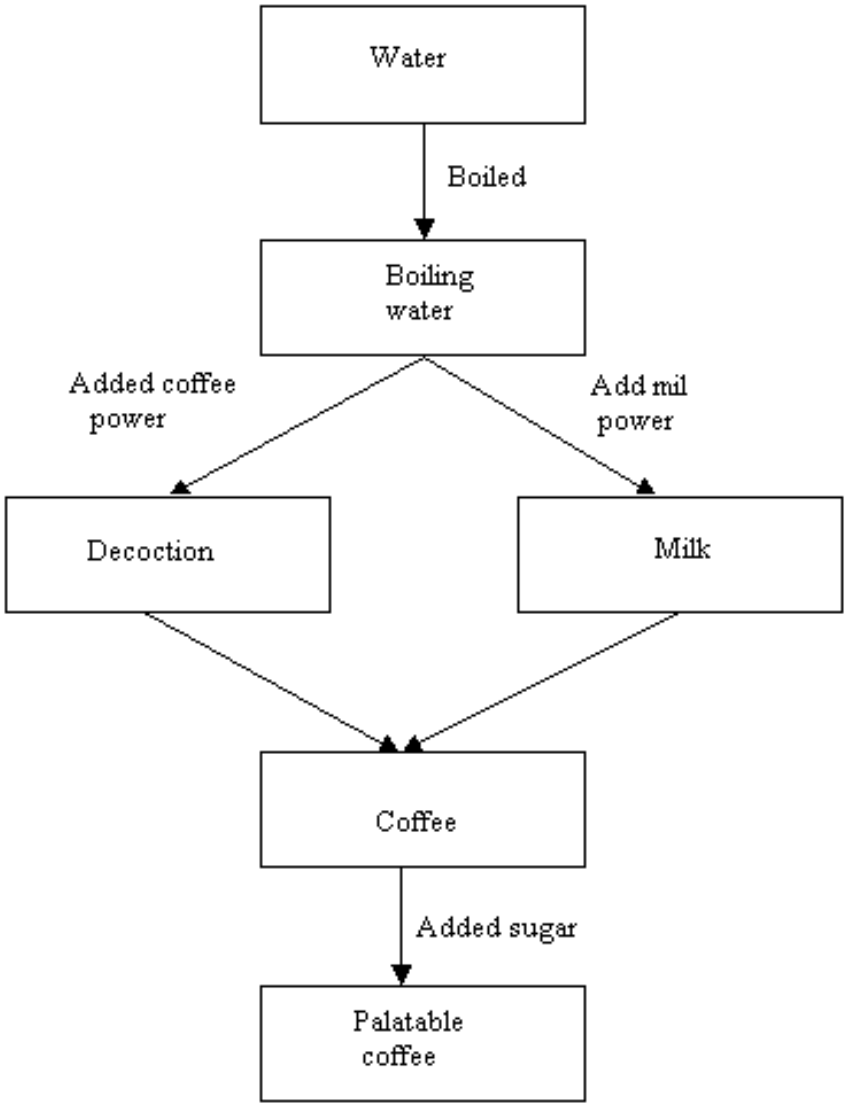
Take same of the boiled water in a cup add necessary amount of instant coffee pour to decoction.

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

Milk decoction and 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, Lob) and loves (Sita, Lob) are called clauses** (i.e. simple sentence/simple provision in law.).

Production system:

Since search forms the core of many intelligent processes, it is useful to structure AI programs in a way that facilities describing of performing search process. A production system provides such system structure as follow:

1. Set of rules as the form
2. One or more KB/Database that contains whatever information is appropriate for particular task
3. Control strategy that specific the order in which the rules will be compared to the database and a way of resolving the conflict that arise when several rules matched at once.
4. Production system is the simplest and one of the oldest techniques for knowledge representation.

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 (is called the working memory)

3. **A control structure/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 cause motion.*
- *The second requirement of a good control strategy is that it cause systematic.*

4. **Rule applier:**

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

- **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.
- **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 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.

Principles of a production system by example: Water Jug Problem

Problem: given two water jugs (4 liter & 3 liter); neither has any measuring markers on it. there is a pump that can use to fill the jugs with water. **How can you get exactly 2 liter water in 4 liter jug?**

Solution:

Let us assume that x & y represents the content of 4L and 3L jugs respectively.

The content of the two jugs will be represented by (x, y). The set of possible PRs are listed below.

1. (x, y) if x < 4 → (4, y) = fill the 4 gallon jug.
2. (x, , y) if y < 3 → (x, 3) = fill the 3 gallon jug.
3. (x, y) if x > 0 → (x-d, y) = pour some water out of the 4 gallon jug.
4. (x, y) if y > 0 → (x, y-d) = pour some water out of the 3 gallon jug.
5. (x, y) if x > 0 → (0, y) = empty the 4 gallon jug on the ground.
6. (x, y) if y > 0 → (x, 0) = empty the 3 gallon jug on the ground.

7. (x, y) if x + y → 4 & y > 0 = (4, y-(4-x)) = pour water from 3L jug into 4L jug until the 4L jug is full.

8.(x, y) if x + y → 3 & x > 0 = (x-(3-y), 3) = pour water from 4L jug into 3L jug until the 3L jug is full.

9.(x, y) if x + y <= 4 & y > 0 → (x + y, 0) = pour all the water from 3L jug to 4L jug.

10.(x, y) if x + y <= 3 & x > 0 → (0, x + y) = pour all the water from 4L jug to 3L jug.

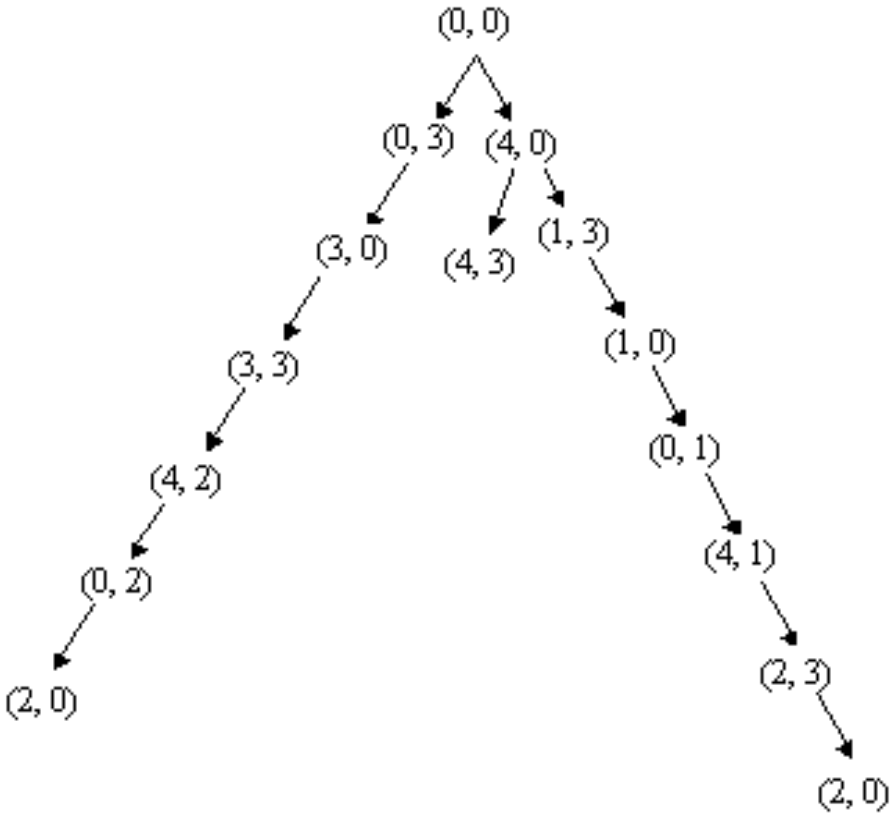
11.(0, 2) = (2, 0) = pour the 2L from 3L jug to 4L jug.

12.(2, y) = (0, y) = empty the 2L in the jug on the ground.

One solution

4L	3L	rule applied
0	0	
		2
0	3	
		9
3	0	
		2
3	3	
		7
4	2	
		5 or 12
0	2	
		9 or 11
2	0	

To keep track of the reasoning process, we draw a state-space for the problem. The leaves generated after firing of the rules should be stored in WM.


(0, 0)
(0, 3)
(4, 0)
(3, 0)
(4, 3)
(1, 3)
(3, 3)
(1, 0)
(4, 2)
(0, 1)
(0, 2)
(4, 1)
(2, 0)
(2, 3)
(2, 0)

Thank You!!!