Production System:
A **production system** (or **production rule system**) is a computer program typically used to provide some form of artificial intelligence, which consists primarily of a set of rules about behavior. These rules, termed **productions**, are a basic representation found useful in automated planning, expert systems and action selection. A production system provides the mechanism necessary to execute productions in order to achieve some goal for the system.

Productions consist of two parts: a sensory precondition (or "IF" statement) and an action (or "THEN"). If a production's precondition matches the current state of the world, then the production is said to be *triggered*. If a production's action is executed, it is said to have *fired*.

A production system also contains a database, sometimes called working memory, which maintains data about current state or knowledge, and a rule interpreter. The rule interpreter must provide a mechanism for prioritizing productions when more than one is triggered.

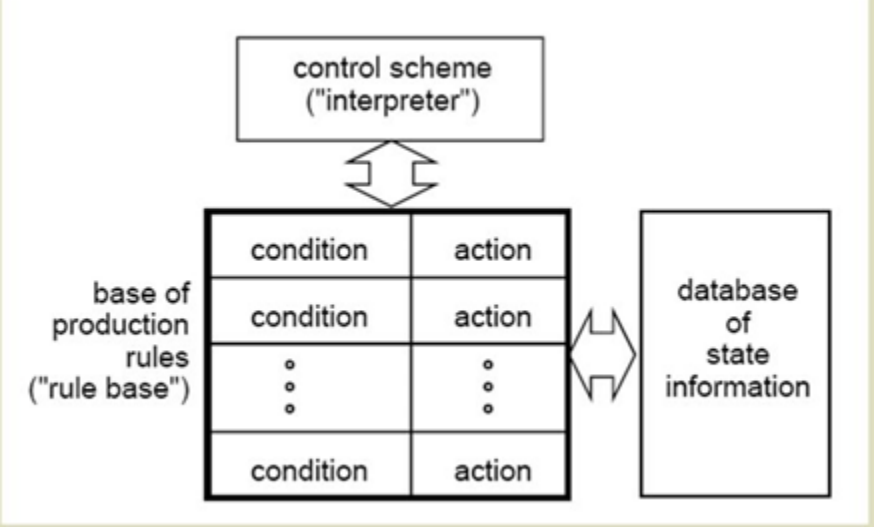
The underlying idea of production systems is to represent knowledge in the form of condition-action pairs called production rules:
If the condition C is satisfied then the action A is appropriate.

Types of production rules

Situation-action rules
If it is raining then open the umbrella.
Inference rules
If Cesar is a man then Cesar is a person

Production system is also called ***ruled-based system***

Architecture of Production System:



Short Term Memory (Database):
- Contains the description of the current state.

Set of Production Rules (Rule Base):
- Set of condition-action pairs and defines a single chunk of problem solving knowledge.

Interpreter:
- A mechanism to examine the short term memory and to determine which rules to fire (According to some strategies such as DFS, BFS, Priority, first-encounter etc)
It checks the condition. If the condition is matched then corresponding actions is taken. If more than 1 rule is matched then such rules are called conflict rules. In such case one of the rule is selected and fired. The process goes on till match is found.

Consider an example:
Problem: Sorting a string composed of letters a, b & c.
Short Term Memory: cbaca
Production Set:
1. ba → ab

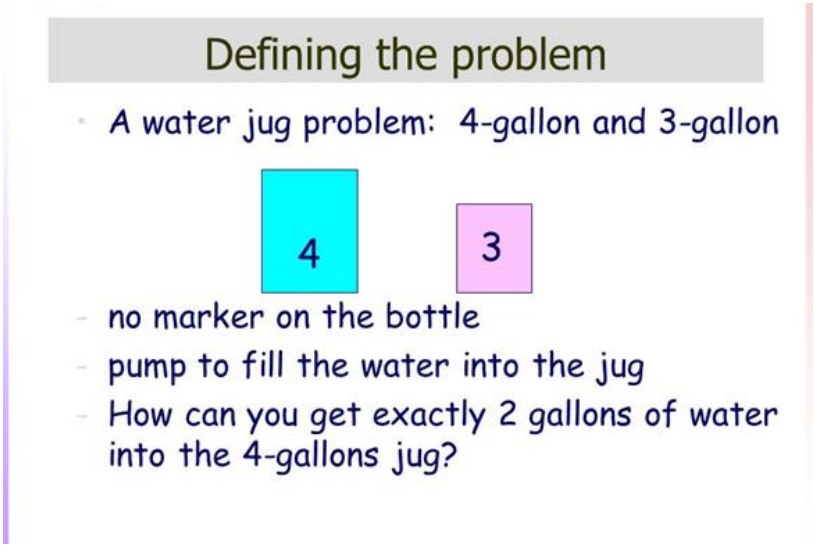
2. ca → ac

3. cb → bc

Interpreter: Choose one rule according to some strategy.

Short memory	Conflict set	Rule fired
cbaca	1,2,3	1
cabca	2	2
acbca	2,3	2
acbac	1,3	1
acabc	2	2
aacbc	3	3
aabcc	0	Halt

The water jug problem
Problem:
There are two jugs, a 4-gallon one and a 3-gallon one. Neither jug has any measuring markers on it. There is a pump that can be used to fill the jugs with water.
How can you get exactly n (0, 1, 2, 3, 4) gallons of water into one of the two jugs ?



Solution Paradigm:

- build a simple production system for solving this problem.
- represent the problem by using the state space paradigm.

State = (x, y); where: x represents the number of gallons in the 4-gallon jug; y represents the number of gallons in the 3-gallon jug. x ε{0, 1, 2, 3, 4} and y ε{0, 1, 2, 3}.

The initial state represents the initial content of the two jugs.
For instance, it may be (2, 3), meaning that the 4-gallon jug contains 2 gallons of water and the 3-gallon jug contains three gallons of water.
The goal state is the desired content of the two jugs.
The left hand side of a production rule indicates the state in which the rule is applicable and the right hand side indicates the state resulting after the application of the rule.

For instance;
(x, y) such that x < 4 →(4, y) represents the production
If the 4-gallon jug is not full then fill it from the pump.

Production rules in water jug problem

1	(x,y) is 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 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 and y>0 →(4,y-(4-x))	Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full
8	(x,y) if X+Y>=3 and x>0→(x-(3-y),3))	Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full.
9	(x,y) if X+Y <=4 and y>0 → (x+y,0)	Pour all the water from the 3-gallon jug into the 4-gallon jug.
10	(x,y) if X+Y<=3 and x>0 →(0,x+y)	Pour all the water from the 4-gallon jug into the 3-gallon jug.
11	(0,2) →(2,0)	Pour the 2-gallon water from 3-gallon jug into the 4-gallon jug.
12	(2,Y)→(0,y)	Empty the 2-gallon in the 4-gallon jug on the ground.

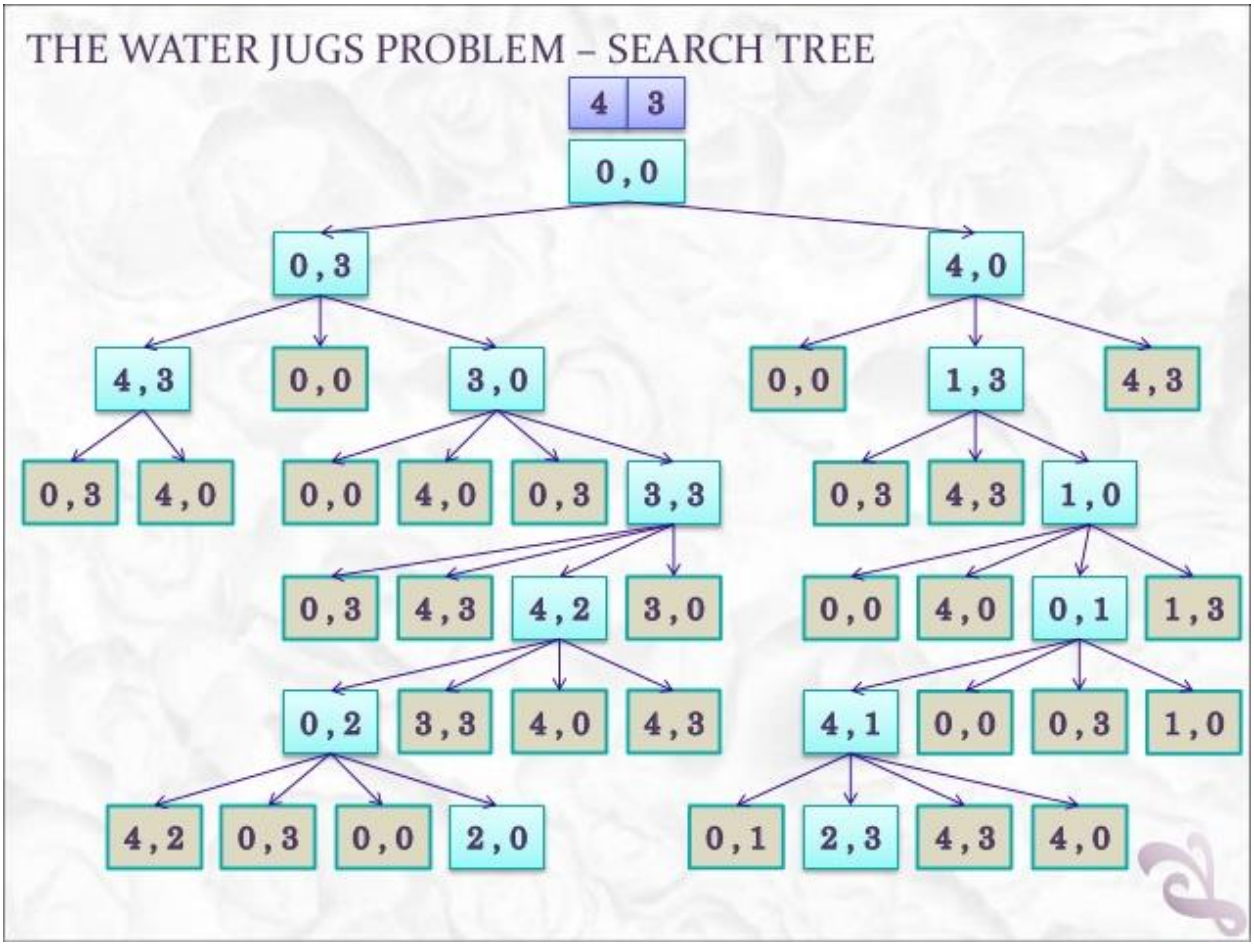
Here rules applied are:

2. Fill the 3 gallon jug

9. Pour all the water from 3 gallon jug to 4 gallon jug

2. Fill the 3 gallon jug

7. Pour the water from 3 gallon jug into 4 gallon jug until 4 gallon jug is full



Major advantages of production systems for artificial intelligence

Separation of knowledge and control

A natural mapping onto state space search

Modularity of production rules

Pattern-directed control

Opportunities for heuristic control of search

Tracing and explanation

Language independence

A plausible model of human problem solving

19

Conflict resolution strategies

Conflict resolution strategies are used in production systems in artificial intelligence, such as in rule-based expert systems, to help in choosing which production rule to fire. The need for such a strategy arises when the conditions of two or more rules are satisfied by the currently known facts.

Conflict resolution strategies fall into several main categories. They each have advantages which form their rationales.

1. Specificity - If all of the conditions of two or more rules are satisfied, choose the rule according to how specific its conditions are. It is possible to favor either the more general or the more specific case. The most specific may be identified roughly as the one having the greatest number of preconditions. This usefully catches exceptions and other special cases before firing the more general (default) rules.
2. Recency - When two or more rules could be chosen, favor the one that matches the most recently added facts, as these are most likely to describe the current situation.
3. Not previously used - If a rule's conditions are satisfied, but previously the same rule has been satisfied by the same facts, ignore the rule. This helps to prevent the system from entering infinite loops.
4. Order - Pick the first applicable rule in order of presentation. This is the strategy that Prolog interpreters use by default, but any strategy may be implemented by building suitable rules in a Prolog system.
5. Arbitrary choice - Pick a rule at random. This has the merit of being simple to compute

Forward chaining and Backward Chaining

Already discussed in unit 4.