101009/IT600C
ARTIFICIAL INTELLIGENCE

Dr. Ranju S Kartha

Associate Professor, DIT

RSET, Cochin

Course Outcomes

After the completion of the course the student will be able to

CO 1: Discuss the fundamental foundations of artificial intelligence (AI).

CO 2: Demonstrate various search methods to solve AI application problems.

CO 3: Develop intelligent algorithms for constraint satisfaction problems.

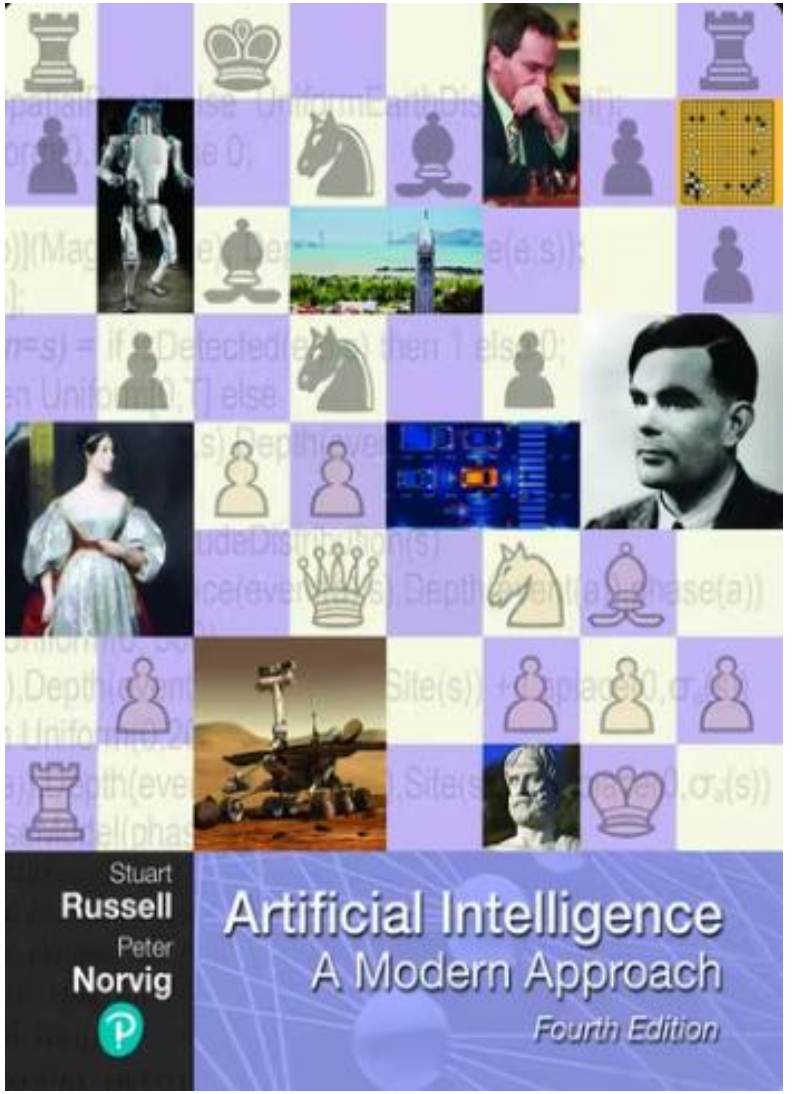
CO 4: Illustrate the concepts of knowledge representation through logics, inference rules and deduce solutions using the principle of resolution.

CO 5: Analyze probabilistic reasoning under uncertainty and the concepts of expert systems.

Text Books

- **Stuart J. Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 4th edition, Pearson, 2021.**

- **Elaine Rich, Kevin Knight and Shivasankar B. Nair, Artificial Intelligence, 3rd Edition, The McGrawHill publications,2009.**



Module 1

Module 1: Introduction, Overview of Artificial intelligence

Problems of AI, AI technique, Tic - Tac - Toe problem. Intelligent Agents, Agents & environment, nature of environment, structure of agents, goal based agents, utility based agents, learning agents.

Problem Solving, Problems, Problem Space & search: Defining the problem as state space search, production system, problem characteristics, issues in the design of search programs.

Definition

- Artificial intelligence is a wide-ranging branch of computer science concerned with building smart machines capable of performing tasks that typically require human intelligence.

- According to the father of Artificial Intelligence, John McCarthy, itis “The science and engineering of making intelligent machines, especially intelligent computer programs”.

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Some definitions of artificial intelligence, organized into four categories.

Thinking humanly: The cognitive modeling approach

- Goal is to build systems that function interally in some way similar to human mind.
- The interdisciplinary field of cognitive science brings together computer models from AI and experimental techniques from psychology to construct precise and testable theories of the human mind.
- Real cognitive science, based on experimental investigation of actual humans.
- Cognitive modeling approach tries to act intelligently while actually internally doing something similar to human mind.

Acting humanly: The Turing Test approach

- The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence.
- The computer would need to possess the following capabilities:
  - Natural language processing to enable it to communicate successfully in English;
  - Knowledge representation to store what it knows or hears;
  - Automated reasoning to use the stored information to answer questions and to draw new conclusions;
  - Machine learning to adapt to new circumstances and to detect and extrapolate patterns

Thinking rationally: The “laws of thought” approach

- Approach firmly grounded in logic.
- For example,
  - “Socrates is a man; all men are mortal; therefore, Socrates is mortal.”
- There are two main obstacles to this approach.
  - First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain.
  - Second, there is a big difference between solving a problem “in principle” and solving it in practice.

Acting rationally:The rational agent approach

- Agents: something that acts i.e. perform some action.
- A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.
- rather than imitating humans trying to solve hard problems, just try to solve hard problems.
- The rational-agent approach has two advantages over the other approaches.
  - First it is more general than the “laws of thought” approach because correct inference is just one of several possible mechanisms for achieving rationality.
  - Second, it is more amenable to scientific development than are approaches based on human behavior or human thought.

History of AI

- The generation of artificial intelligence (1943–1955)
  - Year 1943: Warren McCulloch and Walter pits proposed a model of artificial neurons.
  - Year 1949: Donald Hebb demonstrated an updating rule for modifying the connection strength between neurons- Hebbian learning.
  - Year 1950: Alan Turing proposed a test. The test can check the machine's ability to exhibit intelligent behavior equivalent to human intelligence, called a Turing test.

History of AI

- The birth of artificial intelligence (1956)
  - Year 1955: Allen Newell and Herbert A. Simon created the "first artificial intelligence program"Which was named as "Logic Theorist". This program had proved 38 of 52 Mathematics theorems, and find new and more elegant proofs for some theorems.
  - Year 1956: The word "Artificial Intelligence" first adopted by John McCarthy at the Dartmouth Conference. For the first time, AI coined as an academic field.

History of AI

- Early enthusiasm, great expectations (1952–1969)
  - Year 1966: The researchers emphasized developing algorithms which can solve mathematical problems. Joseph Weizenbaum created the first chatbot in 1966, which was named as ELIZA.
  - Year 1972: The first intelligent humanoid robot was built in Japan which was named as WABOT-1.
- A dose of reality (1966–1973)
  - Realization that many AI problems are intractable.
  - Limitations of existing neural network methods identified

History of AI

- Knowledge-based systems: The key to power? (1969–1979)
  - Development of knowledge-based systems
  - Success of rule-based expert systems, E.g., DENDRAL, MYCIN
  - But were brittle and did not scale well in practice
- AI becomes an industry (1980–present)
  - Year 1980: Expert systems were programmed that emulate the decision-making ability of a human expert.

History of AI

- The return of neural networks (1986–present)
  - Neural networks return to popularity
  - Major advances in machine learning algorithms and applications
  - Bayesian networks as a knowledge representation framework.
- The emergence of intelligent agents (1995–present)
  - Year 1997: IBM Deep Blue beats world chess champion, Gary Kasparov, and became the first computer to beat a world chess champion.
  - Year 2002: for the first time, AI entered the home in the form of Roomba, a vacuum cleaner.
  - Year 2006: AI came in the Business world. Companies like Facebook, Twitter, and Netflix also started using AI

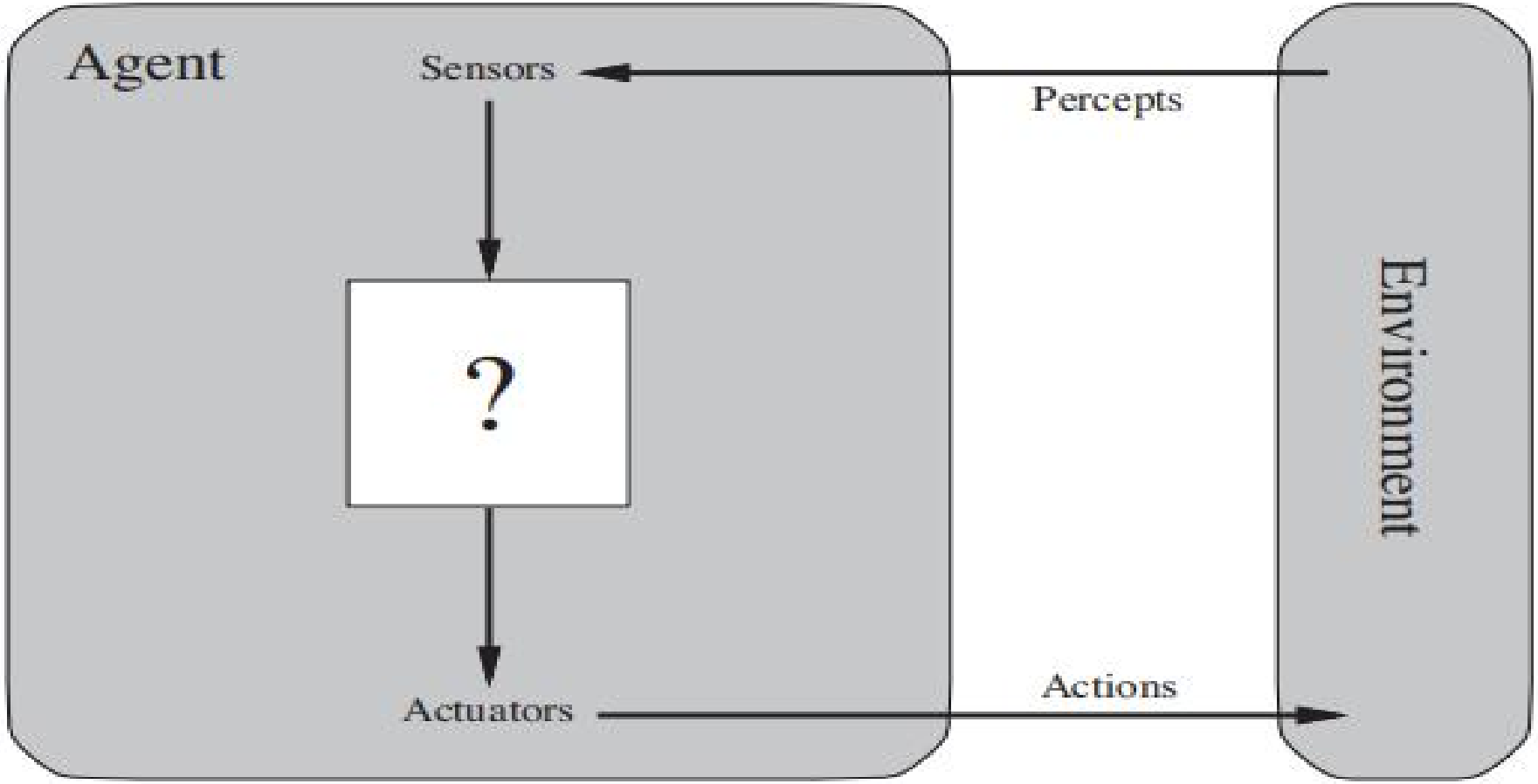
History of AI

- The availability of very large data sets (2001–present)
- 2011–present: Deep learning, big data and artificial general intelligence
  - Year 2011: IBM's Watson won jeopardy, a quiz show, where it had to solve the complex questions as well as riddles. Watson had proved that it could understand natural language and can solve tricky questions quickly.
  - Year 2012: Google has launched an Android app feature "Google now", which was able to provide information to the user as a prediction.
  - Year 2014: Chatbot "Eugene Goostman" won a competition.
  - Year 2018: The "Project Debater" from IBM debated on complex topics with two master debaters and also performed extremely well.

AGENTS AND ENVIRONMENTS

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- An agent’s percept sequence is the complete history of everything the agent has ever perceived.
- In general, an agent’s choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn’t perceived.

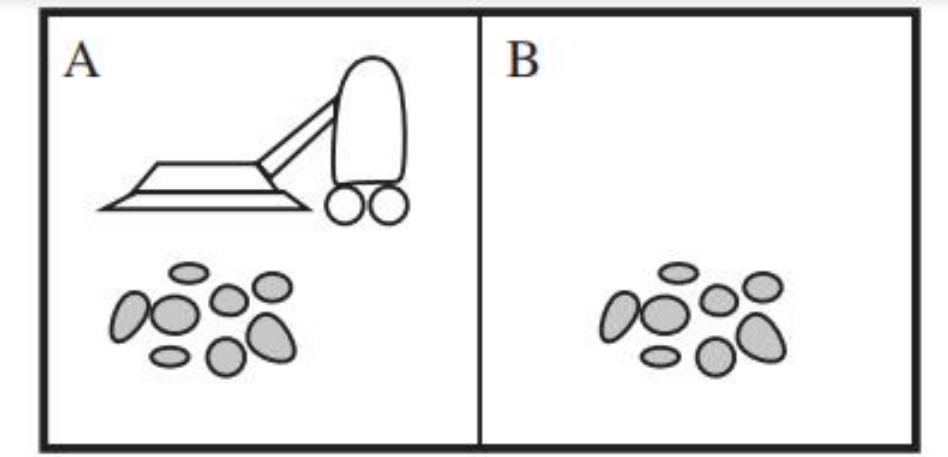
Agent Structure



Agents interact with environments through sensors and actuators.

Agent function

- An agent's behavior is described by the agent function that maps any given percept sequence to an action.
- The agent function for an artificial agent will be implemented by an agent program.
- The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.



A vacuum-cleaner world with just two locations.

Percept Sequence

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

Partial tabulation of a simple agent function for the vacuum-cleaner world

Rational agent

- A rational agent is one that does the right thing—conceptually speaking, every entry in the table for the agent function is filled out correctly.

- For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Nature of the environment

- What is rational at any given time depends on four things:
  - The performance measure that defines the criterion of success.
  - The agent's prior knowledge of the environment.
  - The actions that the agent can perform.
  - The agent's percept sequence to date.
- PEAS (Performance, Environment, Actuators, Sensors)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

PEAS description of the task environment for an automated taxi.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Examples of agent types and their PEAS descriptions.

Structure of Agents

- The job of AI is to design an agent program that implements the agent function— the mapping from percepts to actions.

- We assume this program will run on some sort of computing device with physical sensors and actuators, we call this the architecture:

  agent = architecture + program

Agent programs

- Simple reflex agents - These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- Model-based reflex agents - the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.
- Goal-based agents - The agent program can combine this with the model (the same information as was used in the model based reflex agent) to choose actions that achieve the goal.
- Utility-based agents - An agent’s utility function is essentially an internalization of the performance measure.

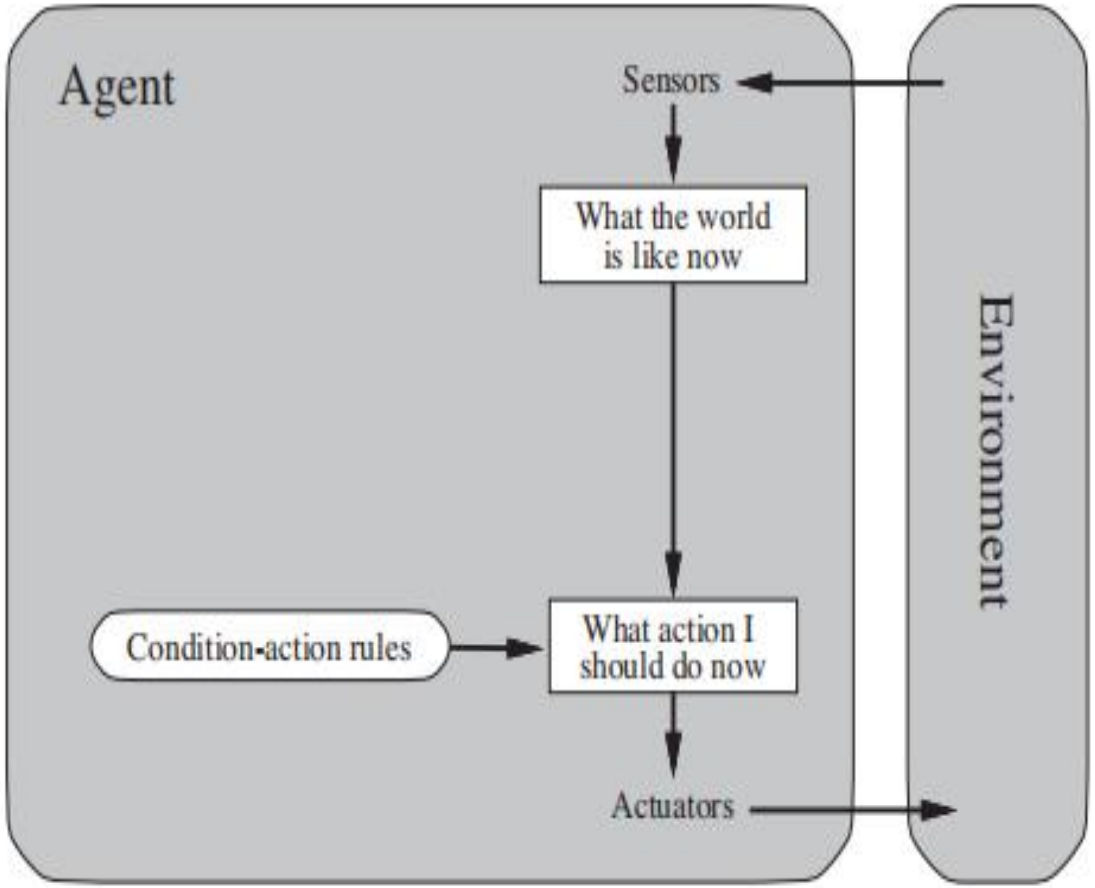
Simple reflex agents

- These agents select actions on the basis of the current percept, ignoring the rest of the percept history.
- Example, the vacuum agent is a simple reflex agent, because its decision is based only on the current location and on whether that location contains dirt.

```
function REFLEX-VACUUM-AGENT([location, status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

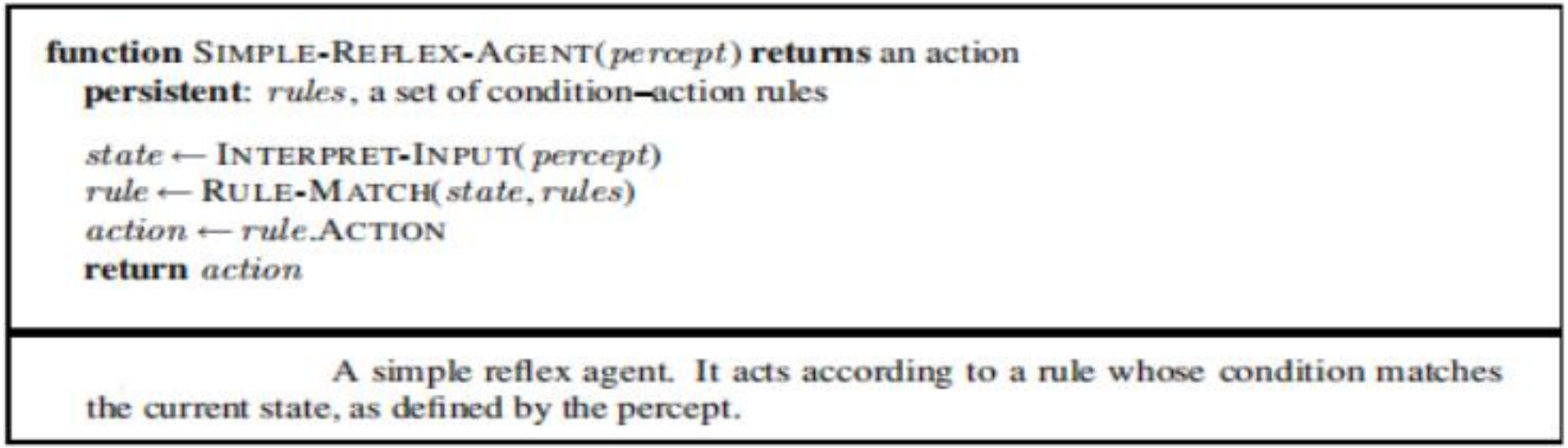
The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function .

Simple reflex agents



- A more general and flexible approach is first to build a general-purpose interpreter for condition– action rules and then to create rule sets for specific task environments.

Agent Program

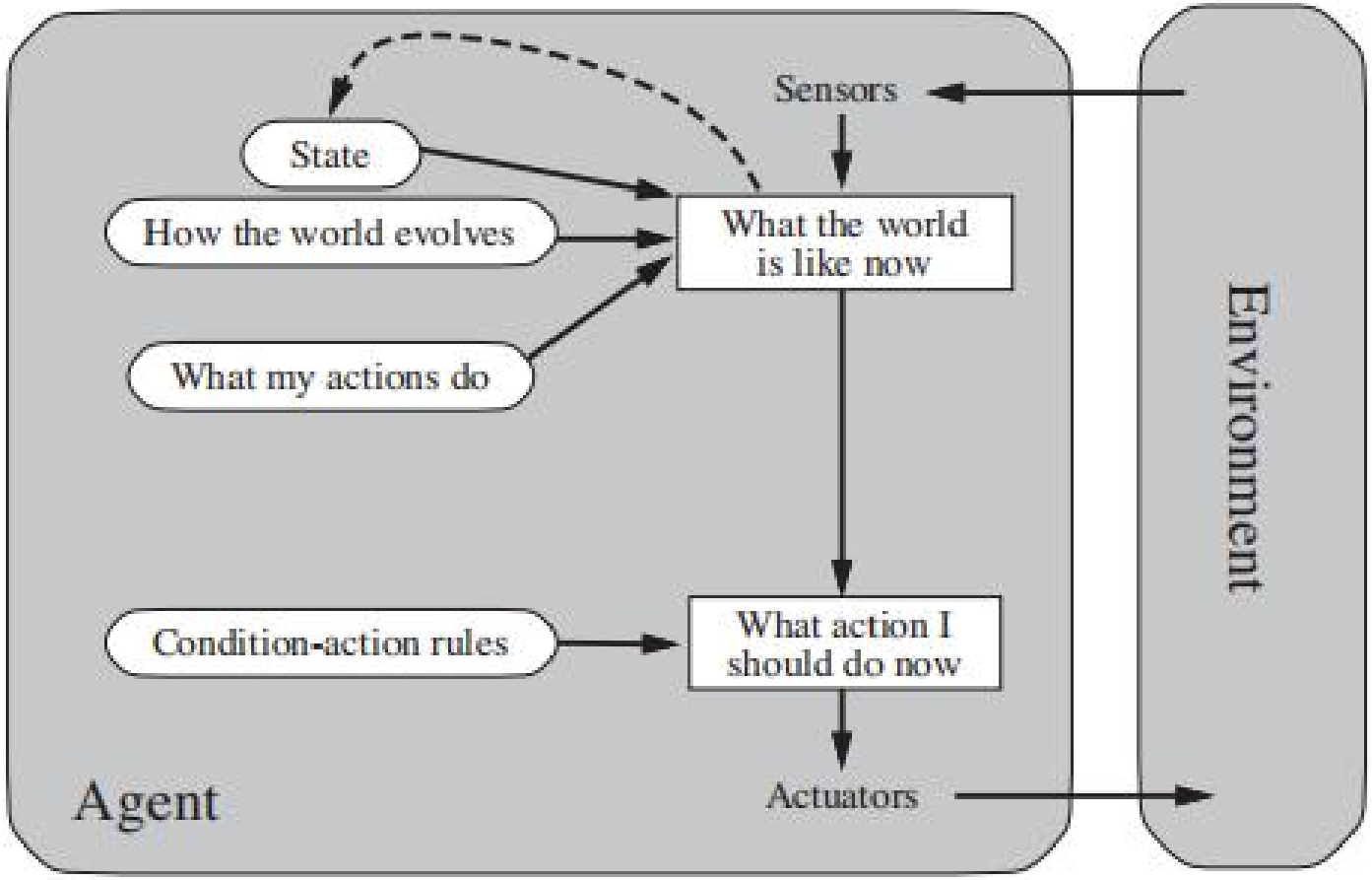


- The INTERPRET-INPUT function generates an abstracted description of the current state from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description.
- The agent will work only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable

Model-based reflex agents

- The most effective way to handle partial observability is for the agent to keep track of the part of the world it can’t see now.
- The agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.

Model-based reflex agents



- The model-based reflex agent with internal state, showing how the current percept is combined with the old internal state to generate the updated description of the current state, based on the agent's model of how the world works.

Agent Program

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

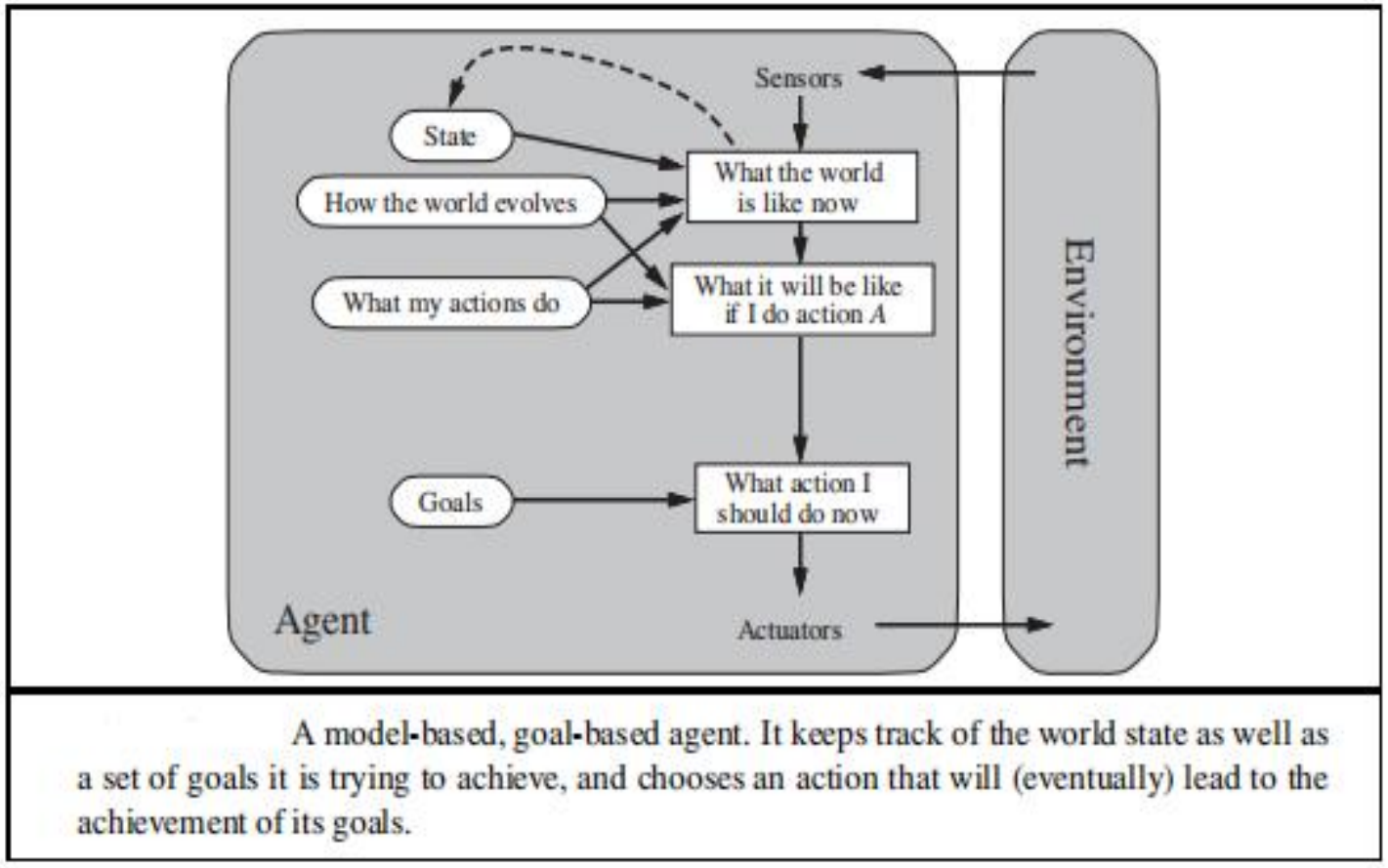
A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

- The function UPDATE-STATE, which is responsible for creating the new internal state description.

Goal-based agents

- The agent needs some sort of goal information that describes GOAL situations that are desirable—for example, being at the passenger's destination.
- Search and planning are thesubfields of AI devoted to finding action sequences that achieve the agent's goals.

Model-based, goal-based agent



A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

- The goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.

Utility-based agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent.
- An agent's utility function is essentially an internalization UTILITY FUNCTION of the performance measure.
- If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

Utility-based agents

- When there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.
- When there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed against the importance of the goals.

Problem-Solving Agents

- Goal-based agent
- Goal formulation - based on the current situation and the agent's performance measure, is the first step in problem solving.
- Problem formulation is the process of deciding what actions and states to consider, given a goal.
- The process of looking for a sequence of actions that reaches the goal is called search.
- A search algorithm takes a problem as input and returns a solution in the form of an action sequence.
- Once a solution is found, the actions it recommends can be carried out. This is called the execution phase.

Well-defined problems and solutions

- A problem can be defined formally by five components:
- The initial state that the agent starts in. For example, the initial state for our agent in Goa might be described as In(Goa).
- A description of the possible actions available to the agent. Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s. We say that each of these actions is applicable in s. For example, from the state In(Goa), the applicable actions are {Go(Delhi), Go(Mumbai)}.

Well-defined problems and solutions

- A description of what each action does; the formal name for this is the transition model, specified by a function RESULT(s, a) that returns the state that results from doing action a in state s.
  - successor - any state reachable from a given state by a single action. For example, we have
- RESULT(In(Goa),Go(Delhi)) = In(Delhi) .
- Together, the initial state, actions, and transition model implicitly define the state space of the problem—the set of all states reachable from the initial state by any sequence of actions.
- The state space forms a directed network or graph in which the nodes are states and the links between nodes are actions.
- A path in the state space is a sequence of states connected by a sequence of actions.

Well-defined problems and solutions

- The **goal test**, which determines whether a given state is a goal state. Sometimes there is an explicit set of possible goal states, and the test simply checks whether the given state is one of them.
- A **path cost function** that assigns a numeric cost to each path. The problem-solving agent chooses a cost function that reflects its own performance measure.
- A **solution** to a problem is an action sequence that leads from the initial state to a goal state. Solution quality is measured by the path cost function, and an optimal solution has the lowest path cost among all solutions.

Problem-Solving Agent

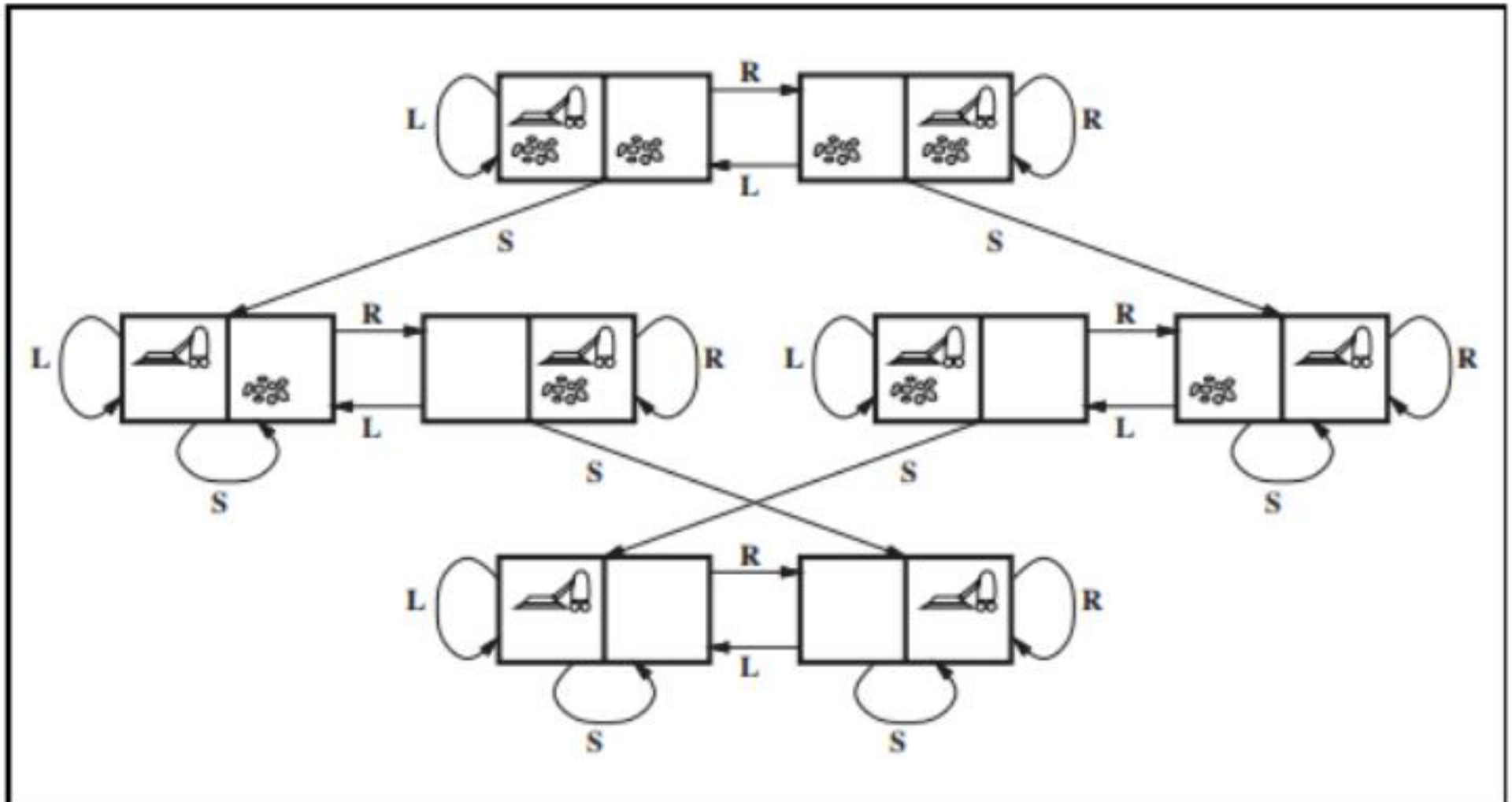
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

A simple problem-solving agent. It first formulates a goal and a problem, searches for a sequence of actions that would solve the problem, and then executes the actions one at a time. When this is complete, it formulates another goal and starts over.

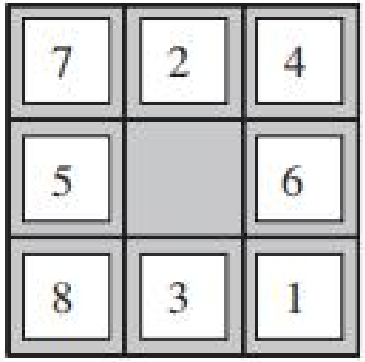
Toy problems: Vacuum world.

- States: The state is determined by both the agent location and the dirt locations. The agent is in one of two locations, each of which might or might not contain dirt. Thus, there are $2 \times 2^2 = 8$ possible world states. A larger environment with n locations has $n \cdot 2^n$ states.
- Initial state: Any state can be designated as the initial state.
- Actions: In this simple environment, each state has just three actions: Left, Right, and
- Suc: Larger environments might also include Up and Down.
- Transition model: The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and Sucking in a clean square have no effect.
- Goal test: This checks whether all the squares are clean.
- Path cost: Each step costs 1, so the path cost is the number of steps in the path.

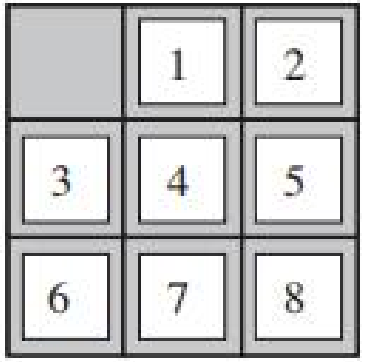


The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

8-puzzle Problem



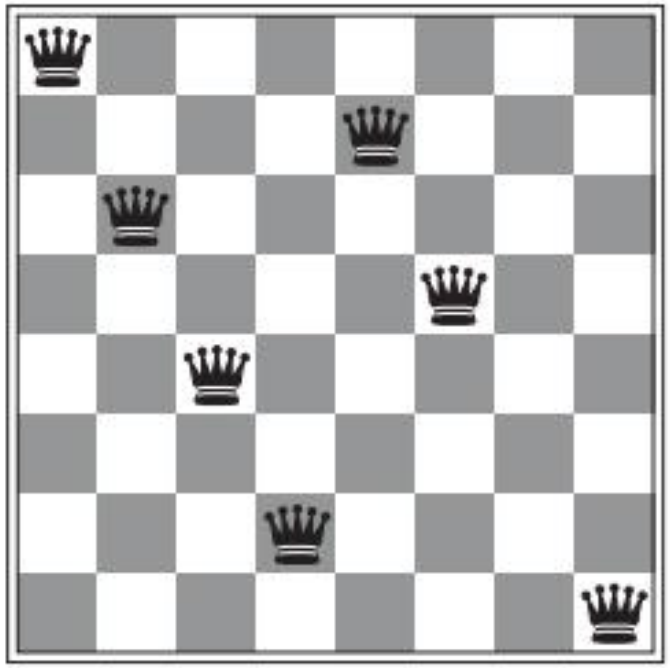
Start State



Goal State

- States: A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- Initial state: Any state can be designated as the initial state.
- Actions: The simplest formulation defines the actions as movements of the blank space Left, Right, Up, or Down. Different subsets of these are possible depending on where the blank is.
- Transition model: Given a state and action, this returns the resulting state
- Goal test: This checks whether the state matches the goal configuration shown in Fig.
- Path cost: Each step costs 1, so the path cost is the number of steps in the path.
- The 8-puzzle has 9!/2 = 181,440 reachable states

8-queens problem



- States: Any arrangement of 0 to 8 queens on the board is a state.
- Initial state: No queens on the board.
- Actions: Add a queen to any empty square.
- Transition model: Returns the board with a queen added to the specified square.
- Goal test: 8 queens are on the board, none attacked.
- In this formulation, we have $64 \cdot 63 \cdots 57 \approx 1.8 \times 1014$ possible sequences to investigate.

State Space Search

- A state space consists of:

1. A representation of the states the system can be in. For example, in a board game, the board represents the current state of the game.
2. A set of operators that can change one state into another state. In a board game, the operators are the legal moves from any given state. Often the operators are represented as programs that change a state representation to represent the new state.
3. An initial state.
4. A set of final states; some of these may be desirable, others undesirable. This set is often represented implicitly by a program that detects terminal states.

The Water Jug Problem

- In this problem, we use two jugs called x and y; x holds a maximum of four gallons of water and y a maximum of three gallons of water. How can we get two gallons of water in the x jug?

- The state space is a set of prearranged pairs giving the number of gallons of water in the pair of jugs at any time, i.e., (x, y) where x = 0, 1, 2, 3 or 4 and y = 0, 1, 2 or 3.
- The start state is (0, 0) and the goal state is (2, n)

Production Rules for the Water Jug Problem)

Sl No	Current state	Next State	Descritpion
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
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

Production Rules for the Water Jug Problem)

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 gallons from 3-gallon jug into the 4-gallon jug
12	(2,y)	(0,y)	Empty the 2 gallons in the 4-gallon jug on the ground

One Solution to the Water Jug Problem

Gallons in the 4-gallon jug	Gallons in the 3-gallon	Rule Applied
0	0	
0	3	2
3	0	9
3	3	2
4	2	7
0	2	5 or 12
2	0	9 or 11

PRODUCTION SYSTEMS

- Production systems provide appropriate structures for performing and describing search processes. A production system has four basic components as enumerated below.
- A set of rules each consisting of a left side that determines the applicability of the rule and a right side that describes the operation to be performed if the rule is applied.
- A database of current facts established during the process of inference.
- A control strategy that specifies the order in which the rules will be compared with facts in the database and also specifies how to resolve conflicts in selection of several rules or selection of more facts.
- A rule firing module

Missionaries and Cannibals Problem

- “Three missionaries and three cannibals are present at one side of a river and need to cross the river. There is only one boat available. At any point of time, the number of cannibals should not outnumber the number of missionaries at that bank. It is also known that only two persons can occupy the boat available at a time.”

Production rules

Rule 1 :	(0, M)	:	One missionary sailing the boat from bank-1 to bank-2
Rule 2 :	(M, 0)	:	One missionary sailing the boat from bank-2 to bank-1
Rule 3 :	(M, M)	:	Two missionaries sailing the boat from bank-1 to bank-2
Rule 4 :	(M, M)	:	Two missionaries sailing the boat from bank-2 to bank-1
Rule 5 :	(M, C)	:	One missionary and one Cannibal sailing the boat from bank-1 to bank-2
Rule 6 :	(C, M)	:	One missionary and one Cannibal sailing the boat from bank-2 to bank-1
Rule 7 :	(C, C)	:	Two Cannibals sailing the boat from bank-1 to bank-2
Rule 8 :	(C, C)	:	Two Cannibals sailing the boat from bank-2 to bank-1
Rule 9 :	(0, C)	:	One Cannibal sailing the boat from bank-1 to bank-2
Rule 10 :	(C, 0)	:	One Cannibal sailing the boat from bank-2 to bank-1

Solution

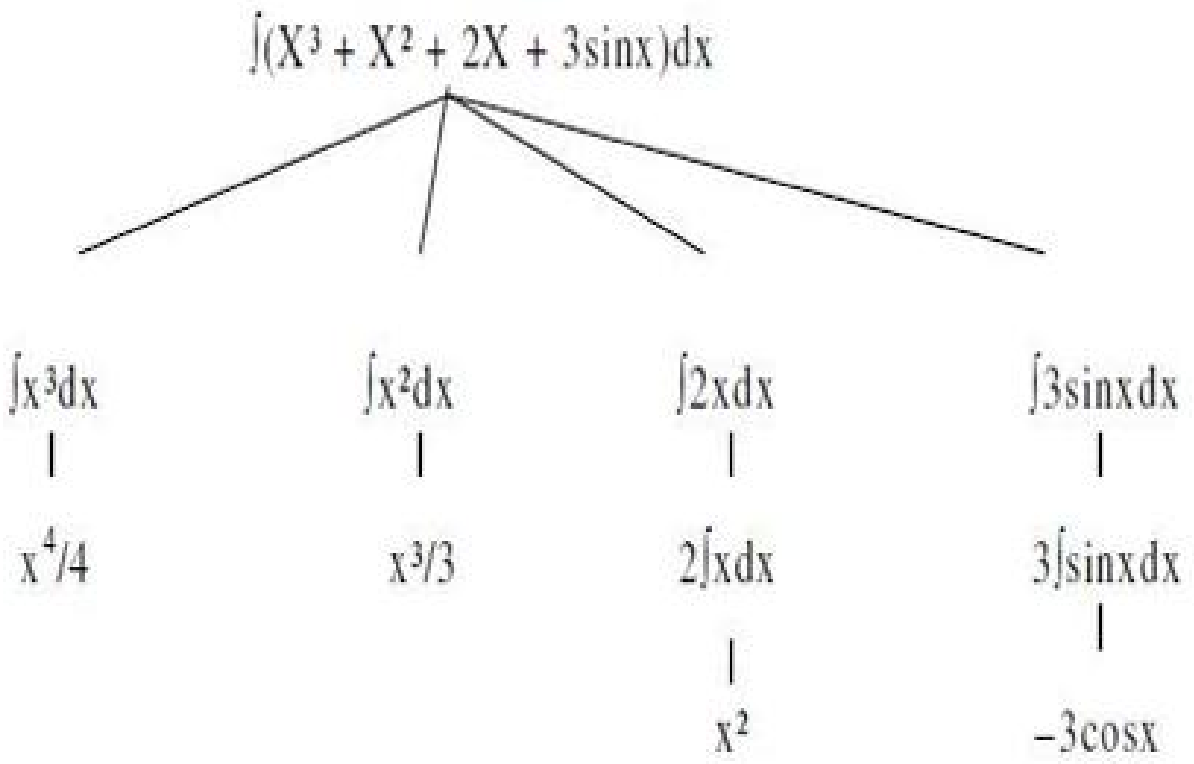
After application of rule	persons in the river bank-1	persons in the river bank-2	boat position
Start state	M, M, M, C, C, C	0	bank-1
5	M, M, C, C	M, C	bank-2
2	M, M, C, C, M	C	bank-1
7	M, M, M	C, C, C	bank-2
10	M, M, M, C	C, C	bank-1
3	M, C	C, C, M, M	bank-2
6	M, C, C, M	C, M	bank-1
3	C, C	C, M, M, M	bank-2
10	C, C, C	M, M, M	bank-1
7	C	M, M, M, C, C	bank-2
10	C, C	M, M, M, C	bank-1
7	0	M, M, M, C, C, C	bank-2

PROBLEM CHARACTERISTICS

- Decomposability of the problem into a set of independent smaller subproblems
- Possibility of undoing solution steps, if they are found to be unwise
- Predictability of the problem universe.
- Possibility of obtaining an obvious solution to a problem without comparison of all other possible solutions
- Type of the solution: whether it is a state or a path to the goal state
- Role of knowledge in problem solving
- Nature of solution process: with or without interacting with the user

Is the problem decomposable?

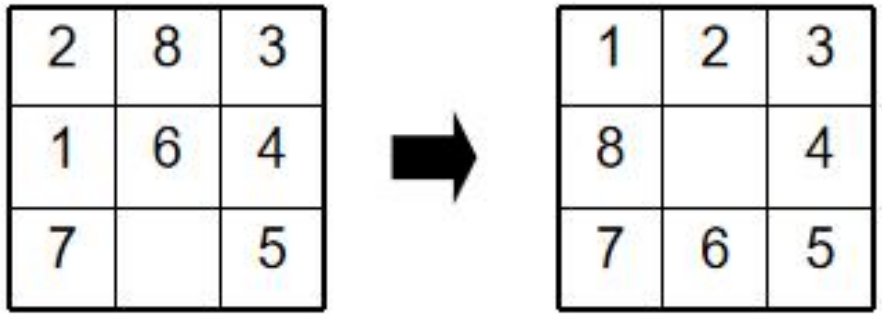
- Can the problem be broken down to smaller problems to be solved independently?
- Decomposable problem can be solved easily.



Can solution steps be ignored or undone?

- The problems are illustrated in the three important classes of problems mentioned below:
- 1. Ignorable, in which solution steps can be ignored. Eg: Theorem Proving
- 2. Recoverable, in which solution steps can be undone. Eg: 8-Puzzle
- 3. Irrecoverable, in which solution steps cannot be undone. Eg: Chess

The 8-Puzzle



Moves can be undone and backtracked.

Recoverable!

Is the universe predictable?

The 8-Puzzle

Every time we make a move, we know exactly what will happen.

Certain outcome!

Playing Bridge

We cannot know exactly where all the cards are or what the other players will do on their turns.

Uncertain outcome!

Is the universe predictable?

- For certain-outcome problems, planning can used to generate a sequence of operators that is guaranteed to lead to a solution.
- For uncertain-outcome problems, a sequence of generated operators can only have a good probability of leading to a solution.
- Plan revision is made as the plan is carried out and the necessary feedback is provided.

Is a good solution absolute or relative?

1. Siva was a man.
2. Siva was a worker in a company.
3. Siva was born in 1905.
4. All men are mortal.
5. All workers in a factory died when there was an accident in 1952.
6. No mortal lives longer than 100 years.

Suppose we ask a question: 'Is Siva alive?'

Is a good solution absolute or relative?

There are two ways to answer the question shown below:

Method I:

1. Siva was a man.
2. Siva was born in 1905.
3. All men are mortal.
4. Now it is 2008, so Siva's age is 103 years.
5. No mortal lives longer than 100 years.

Method II:

1. Siva is a worker in the company.
2. All workers in the company died in 1952.

Answer: So Siva is not alive. It is the answer from the above methods.

Is the solution a state or a path?

Finding a consistent intepretation

"The bank president ate a dish of pasta salad with the fork".

- "bank" refers to a financial situation or to a side of a river?
- "dish" or "pasta salad" was eaten?
- Does "pasta salad" contain pasta, as "dog food" does not contain "dog"?
- Which part of the sentence does "with the fork" modify? What if "with vegetables" is there?

No record of the processing is necessary.

Is the solution a state or a path?

The Water Jug Problem

The path that leads to the goal must be reported.

What is the role of knowledge

Playing Chess

Knowledge is important only to constrain the search for a solution.

Reading Newspaper

Knowledge is required even to be able to recognize a solution.

Does the task require human-interaction?

- Solitary problem, in which there is no intermediate communication and no demand for an explanation of the reasoning process.

- Conversational problem, in which intermediate communication is to provide either additional assistance to the computer or additional information to the user.

CHARACTERISTICS OF PRODUCTION SYSTEMS

- Production systems provide us with good ways of describing the operations that can be performed in a search for a solution to a problem.
- At this time, two questions may arise:
  - 1. Can production systems be described by a set of characteristics? And how can they be easily implemented?
  - 2. What relationships are there between the problem types and the types of production systems well suited for solving the problems?

Classes of production systems

- A monotonic production system is a production system in which the application of a rule never prevents the later application of another rule that could also have been applied at the time the first rule was selected.
- A non-monotonic production system is one in which this is not true.
- A partially commutative production system is a production system with the property that if the application of a particular sequence of rules transforms state P into state Q, then any combination of those rules that is allowable also transforms state P into state Q.
- A commutative production system is a production system that is both monotonic and partially commutative.

Four Categories of Production System

Production System	Monotonic	Non-monotonic
Partially Commutative	Theorem Proving	Robot Navigation
Non-partially Commutative	Chemical Synthesis	Bridge

Issues in the Design of Search Programs

- Each search process can be considered to be a tree traversal.
- The following issues arise when searching:
- The tree can be searched forward from the initial node to the goal state or backwards from the goal state to the initial state.
- To select applicable rules, it is critical to have an efficient procedure for matching rules against states.
- How to represent each node of the search process? This is the knowledge representation problem or the frame problem. In games, an array suffices; in other problems, more complex data structures are needed.

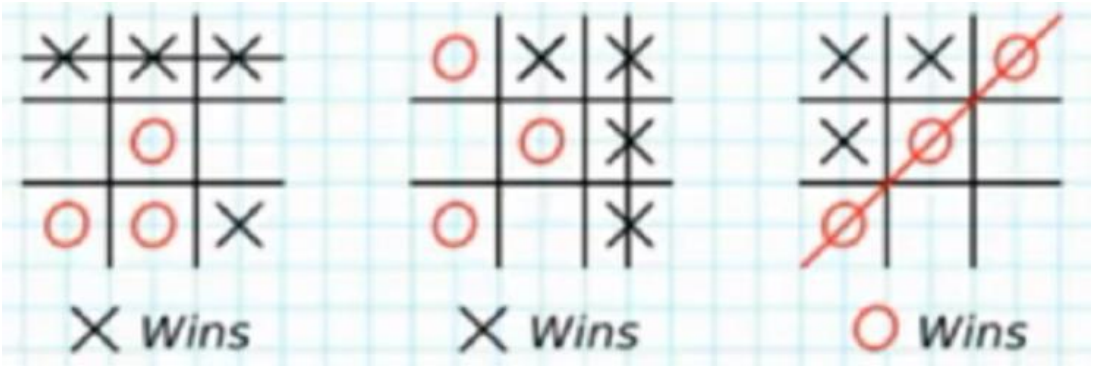
Check duplicate nodes

- 1. Observe all nodes that are already generated, if a new node is present.
- 2. If it exists add it to the graph.
- 3. If it already exists, then
  - a. Set the node that is being expanded to the point to the already existing node corresponding to its successor rather than to the new one. The new one can be thrown away.
  - b. If the best or shortest path is being determined, check to see if this path is better or worse than the old one. If worse, do nothing.
- Better save the new path and work the change in length through the chain of successor nodes if necessary.

Tic-Tac-Toe Problem in AI

- It’s a paper and pencil game of two players X and O, who choose to mark a space on a grid of 3×3.

- The game is won by the player who succeeds in putting three of their marks in a horizontal line, vertical line, or diagonal line.



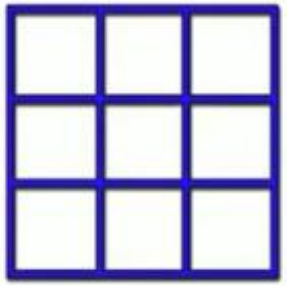
Tic-Tac-Toe Problem in AI

- This game is played by two players – One is a human and the other is a computer.
- The objective is to write a computer program in such a way that the computer wins most of the time.
- Three approaches are presented to play this game which increases in,
  - The complexity of the approach
  - Use of generalization of the approach
  - Clarity of their knowledge
  - Extensibility of their approach

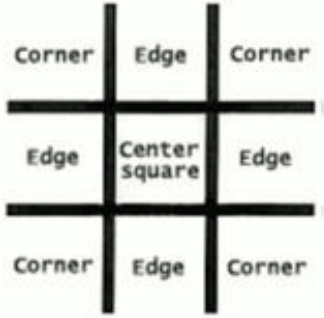
Program 1

- Assume,
- Player 1 – uses X and
- Player 2 – use O
- So, a player who gets 3 consecutive marks first, the player will win the game.

- **Board Data Structure**



The cells could be represented as Center square, Corner, Edge as like below



Each square is numbered, which starts from 1 to 9 like following image

1	2	3
4	5	6
7	8	9

2-D game-board

1	2	3	4	5	6	7	8	9

1D vector

Data Structures: Board and Move Table

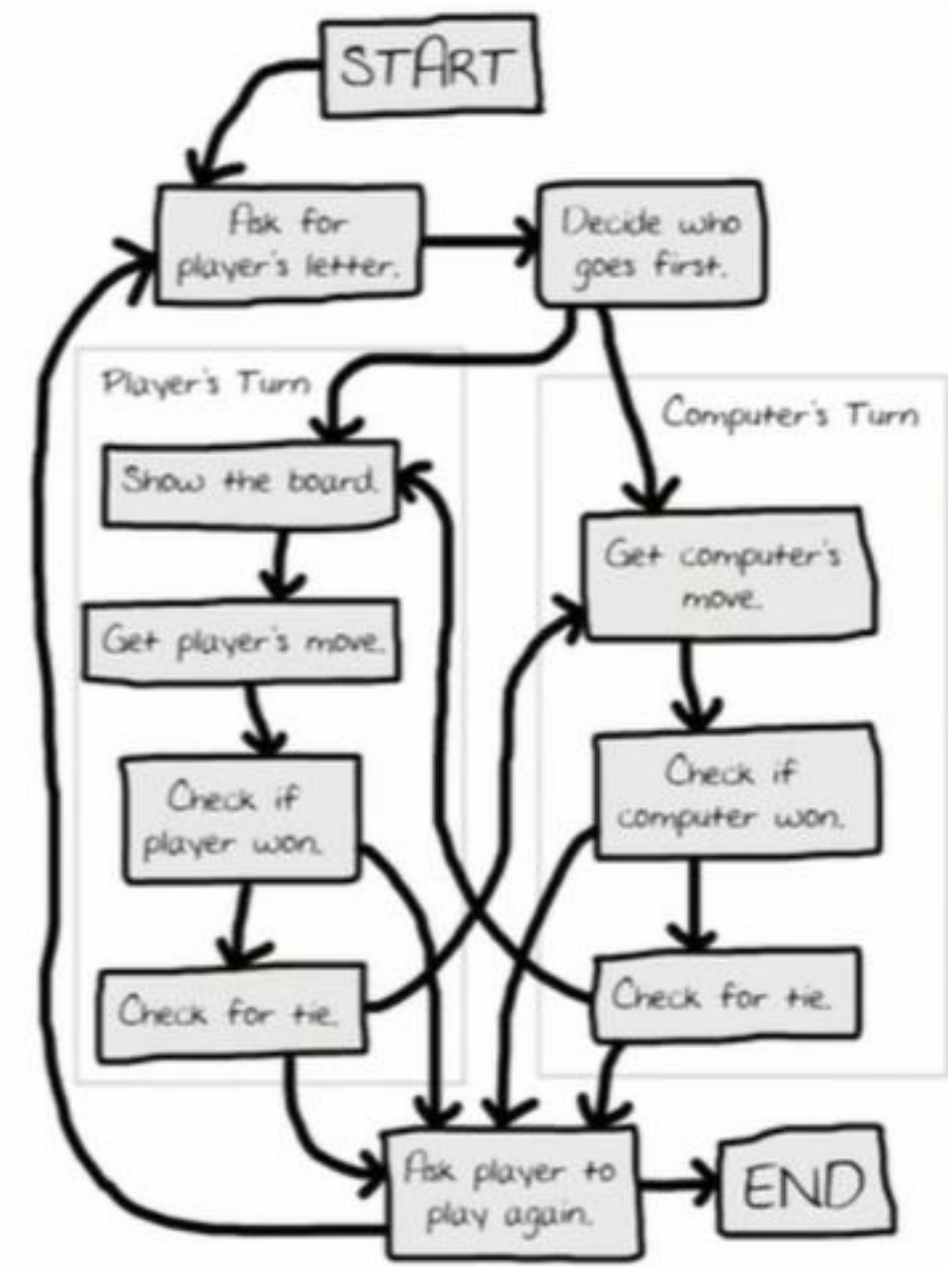
- Consider a Board having nine elements vector. Each element will contain:
- 0 for blank
- 1 indicating X Player move
- 2 indicating O Player move
- Computer may play as X or O Player
- First Player is always X

- **Move Table -** It is a vector of 3^9 elements, each element of which is a 9 element vector representing board position.
- Total of 3^9 (19683) elements in move table

Program 1

Index	Current Board position	New Board position
0	000000000	000010000
1	000000001	020000001
2	000000002	000100002
3	000000010	002000010
...		

Move Table



Program 1 - Advantages and Disadvantages

- Advantages - This program is very efficient in time
- Disadvantages
  - A lot of space to store the Move-Table.
  - A lot of work to specify all the entries in the Move-Table.
  - Difficult to extend
  - Highly error prone as the data is voluminous
  - Poorextensibility
  - Not intelligent at all

Tic-Tac-Toe Problem using Magic Square – Program 2 in AI

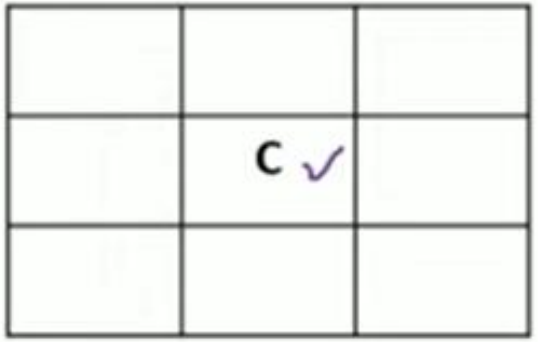
- We assign board position to vector elements
- Sum of all rows,columns and diagonals must be 15

8	3	4
1	5	9
6	7	2

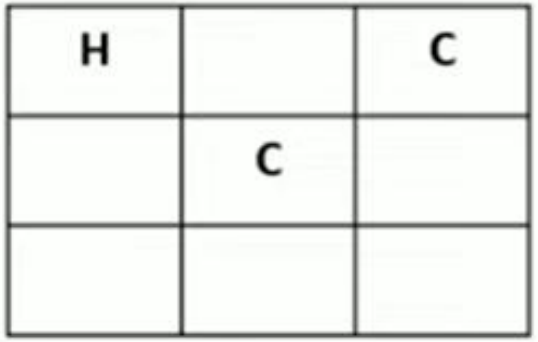
- Algorithm:
- First machine will check chance to win
  - difference between 15 and the sum of the two squares
  - if the difference is not positive or if it is greater than 9, then the original two squares were not collinear and so can be ignored
- or it will check the opponent of winning and block the chances of winning

Example

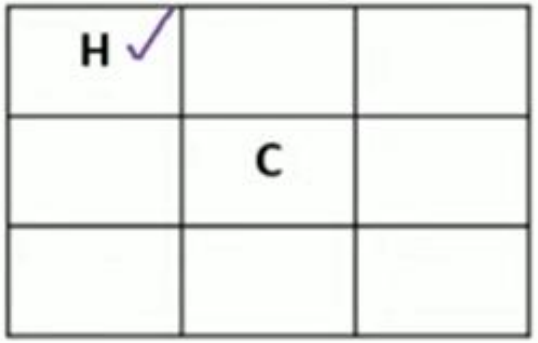
- **Turn – Computer (C)**



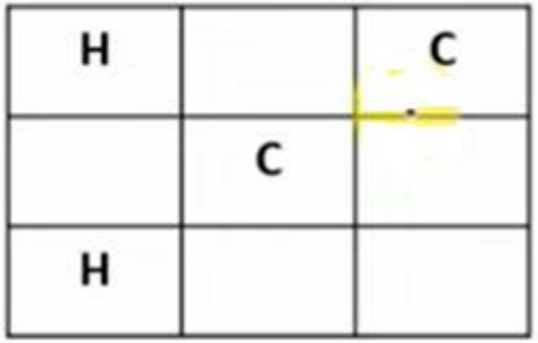
Turn – Computer (C)



- **Turn – Human (H)**



Turn – Human (H)



Example

- **Previous State**

H		C
C	C	
H		

- **Turn – Computer (C)**

H		C
C	C	C
H	H	

Magic Square

8	3	4
1	5	9
6	7	2

- **Turn – Human (H)**

H		C
C	C	
H	H	

Diff = 15 - (5+4) = 6

6 is not empty, hence Computer can't win the game.

Diff = 15 – (1+4) = 10

10 is greater than 9, hence Computer can't win the game.

Diff = 15 – (1+5) = 9

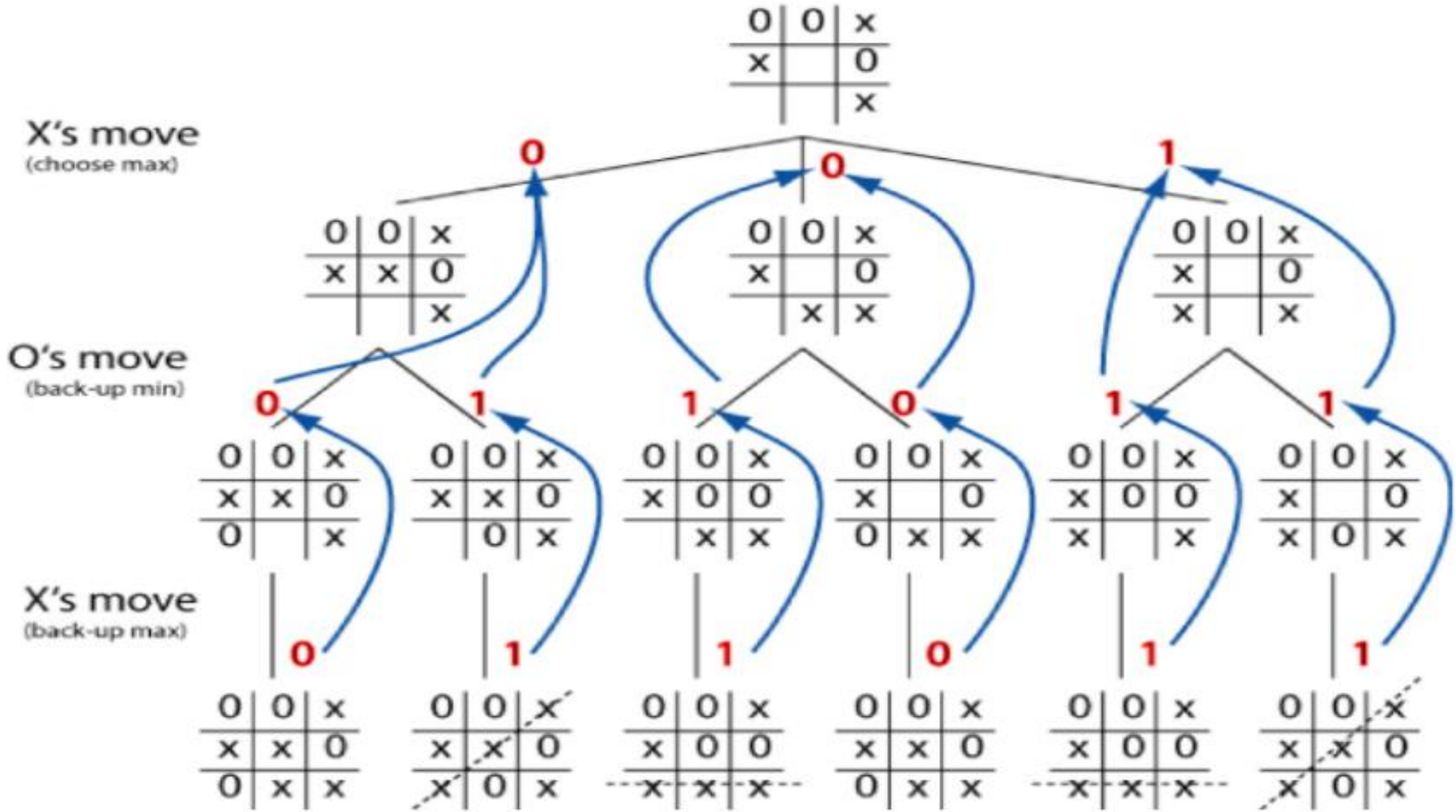
9 is empty, hence Computer can win the game. Computer - go to 9

- The proposed solution is not efficient in terms of time, as it has to check several conditions before making each move.
- Easier to understand the program's strategy.
- Hard to generalize the program or solution

Tic-Tac-Toe Problem – Program 3 in AI

- Follow these steps to select the next move,
- If it is a win, give it the highest rating.
- Otherwise, consider all the moves the opponent could make next. Assume the opponent will make the move that is worst for us. Assign the rating of that move to the current node.
- The best node is then the one with the highest rating.

- Require much more time to consider all possible moves.
- Could be extended to handle more complicated games.


X's move
(choose max)
O's move
(back-up min)
X's move
(back-up max)