ARTIFICIAL INTELLIGENCE

PROF. BHAVISHA SUTHAR

PROBLEMS AND PROBLEM SPACES & SEARCH

- AI Problems
- Underlying Assumptions,
- The Level Of the Model
- Criteria for success
- Defining the Problem as State Space Search
- Production Systems
- Problem Characteristics
- Issues in the Design of Search Programs
- Problems

WHAT IS AI? DEFINITION

- **According to Patterson**, "AI is a branch of computer science that deals with the study and the creation of computer systems that exhibit some form of intelligence.
- Intelligence means that

a. Systems that learn new concepts and tasks
b. System that can reason and draw useful conclusions about the world around us.
c. Systems that can understand a natural language or perceive and comprehend a visual scene,
d. And systems that perform other types of feats that require human types of intelligence.

WHAT IS AI? DEFINITION

- **General definitions:** "An understanding of AI requires an understanding of related terms such as intelligence, knowledge, reasoning, thought, cognition, learning and solving problems."
- **By Advert:** "AI is the part of computer science concerned with designing intelligent computer systems i.e. system that exhibit characteristics that we associate with intelligent in human behavior.
- **By heuristic:** "AI is the branch of computer science that deals with the ways of representing knowledge using symbols rather than numbers and with the rules of thumb for processing information."
- **Modern definition:** "AI is defined as the branch of computer science dealing with symbolic and non-algorithmic method of problem solving."

THEOREM PROVING

- The proving of Mathematical theorems by a computer program.
- Theorems automatically proven from a given set of axioms
- Theorems & axioms expressed in logic and logical inferences applied
- First Theorem prover developed in mid-50s but breakthrough in 1960s
- Breakthrough came after introduction of Resolution of Resolution inference rule
- Example
  - All Irish are Europeans.
  - Dave is a Irish
  - Therefore, Dave is a European

AI TASK DOMAIN

1. Mundane Tasks
2. Formal Tasks
3. Expert Tasks

MUNDANE TASK

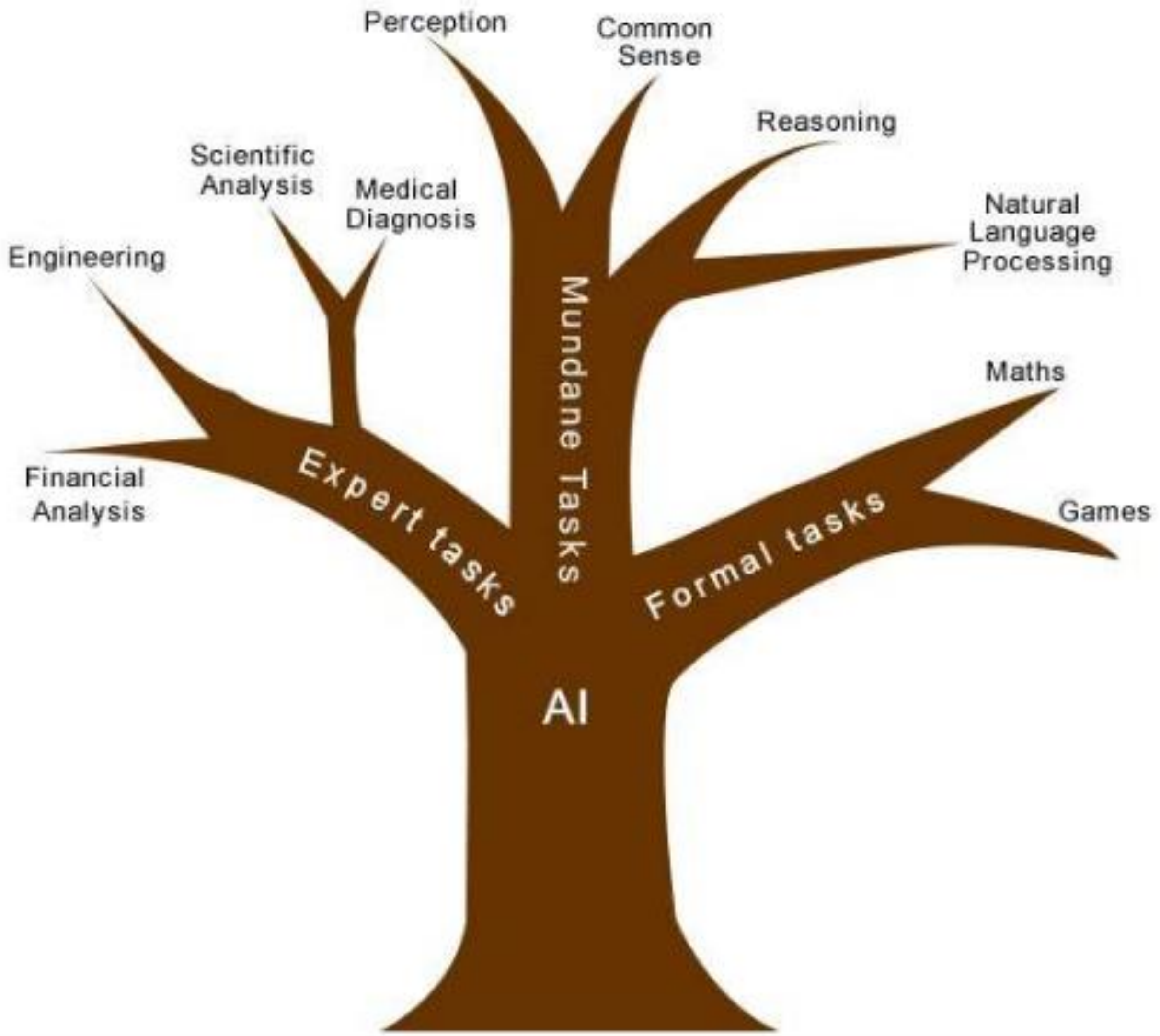
a. Perception of vision and speech
b. Natural Language understanding
c. Common sense Reasoning
d. Robotics

FORMAL TASKS

a. Game Playing, e.g. chess, 8-queens problems, water jug problem etc.
b. Mathematics- geometry, calculus etc.

EXPERT TASKS

- Engineering field
- Scientific analysis
- Medical and financial analysis

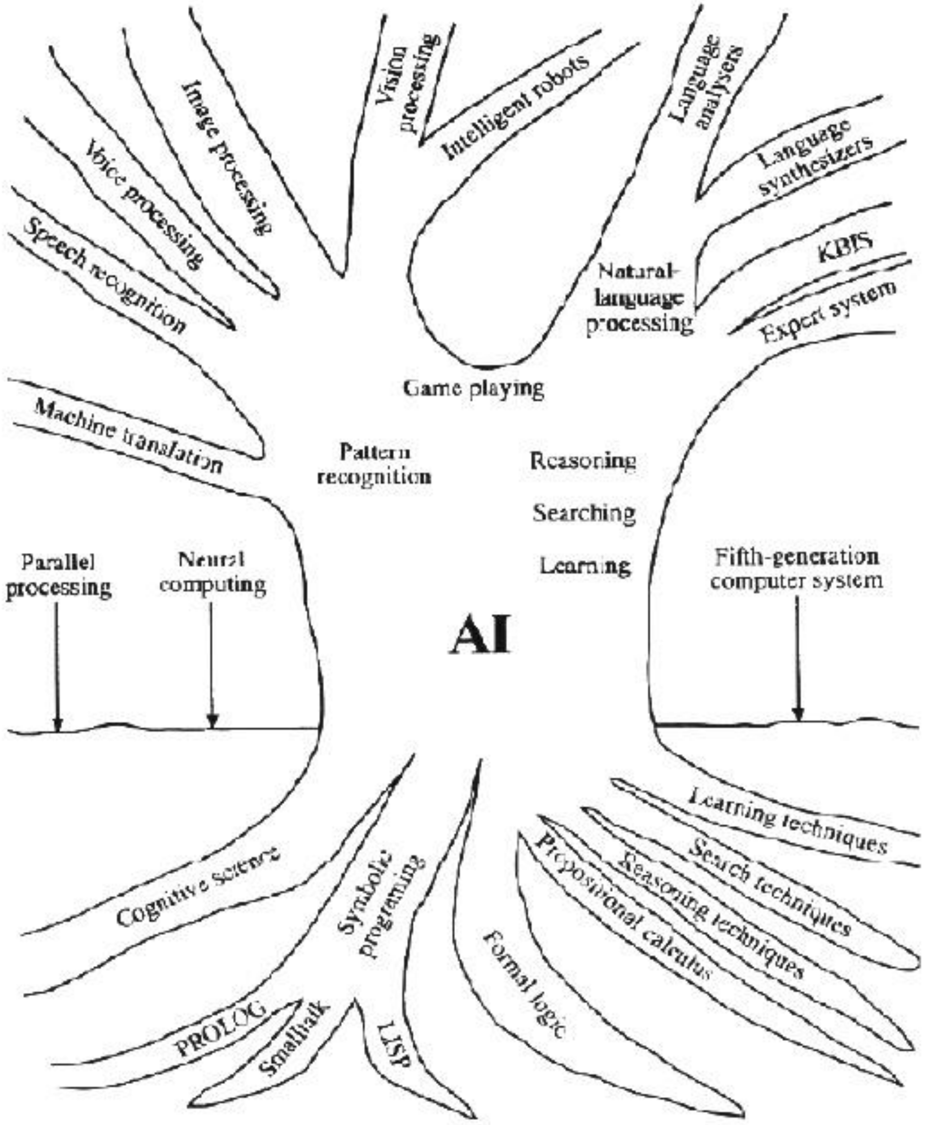


Task Domains of Artificial Intelligence

AREAS OF AI

- Robotics
- Memory Organization
- Knowledge representation
- Storage and recall
- Learning models
- Inference Techniques
- Common sense reasoning
- Decision Making
- Pattern Recognition
- Searching
- Speech recognition
- Speech synthesis- translating speech into audio

(5) Other AI fields - a tree representation



SCOPE OF AI

- Games
- Theorem Proving
- Natural language Processing
- Vision and Speech Processing
- Robotics
- Expert System

AI TECHNIQUES (SEARCH KNOWLEDGE)

- Knowledge can be defined as the body of facts and principles accumulated by humankind or the act, fact or state of knowing.
  - Example, knowledge stored in complex structures of interconnected neurons. The structures correspond to symbolic representations of the knowledge possessed by the organisms, the facts, rules, and so on.
- Types:
  1. Procedural or Operational Knowledge
     - Steps to solve quadratic equation
  2. Declarative or Relational Knowledge
     - Facts about the world
  3. Heuristic Knowledge
     - Tricks, strategy, experience to simplify the solution to problems.

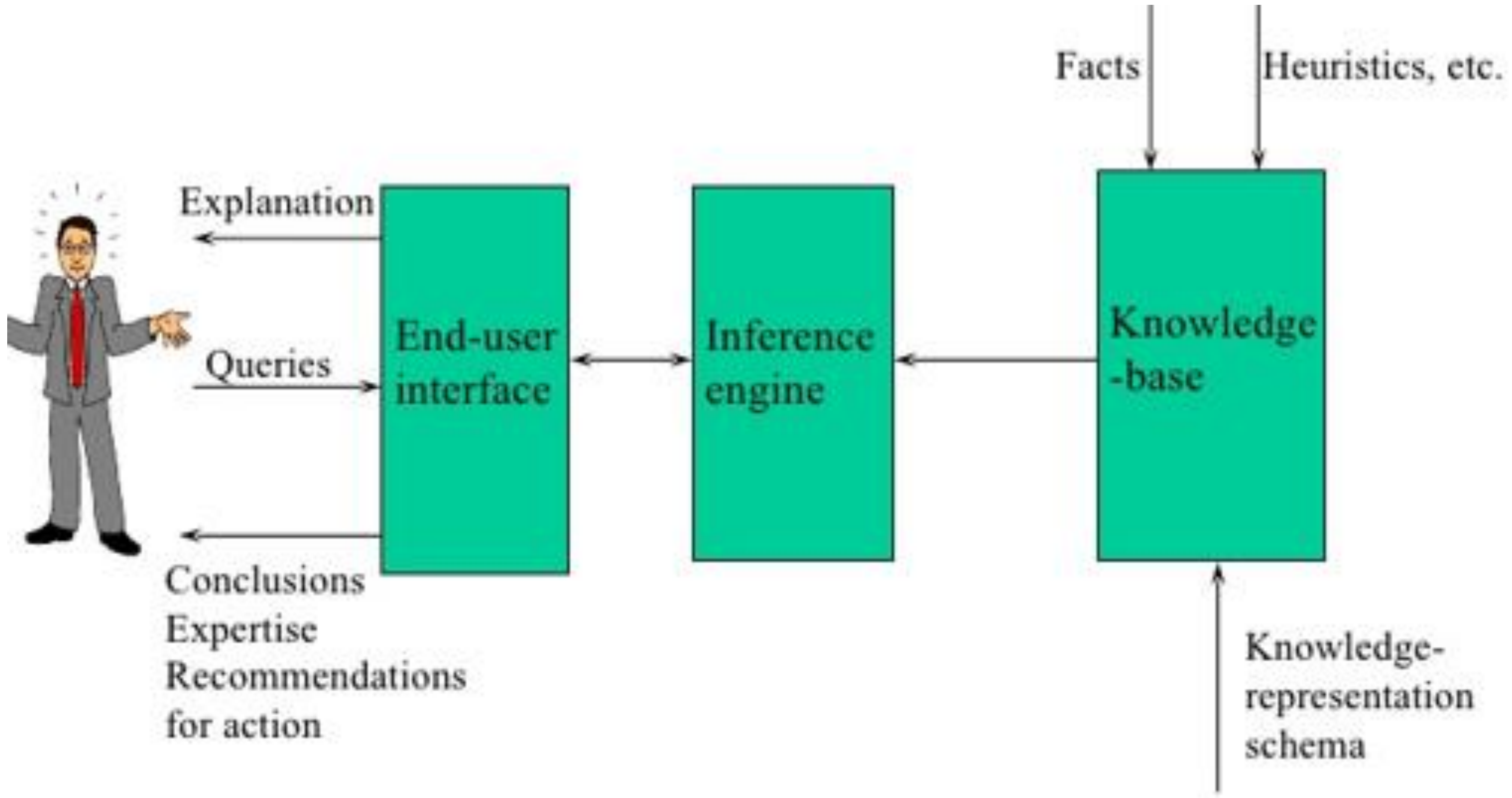
KNOWLEDGE TERMINOLOGY

- **Knowledge and data:** A Doctor has both. Data is patient's record whereas knowledge is what he has learned in his medical college
- **Belief:** meaningful and coherent expression that can be represented.
- **Hypothesis:** a Justified Belief that is not known to be true
- **Knowledge:** true justified belief.
- **Meta Knowledge:** knowledge about knowledge
- **Epistemology:** study of the nature of knowledge

KNOWLEDGE BASED SYSTEM (KBS)

- Those systems that depend on a rich base of knowledge to perform difficult tasks are known as Knowledge based Systems
- Three main Components of KBS
  1. Input-Output Unit
  2. Inference Control Unit
  3. Knowledge Base

KBS ARCHITECTURE



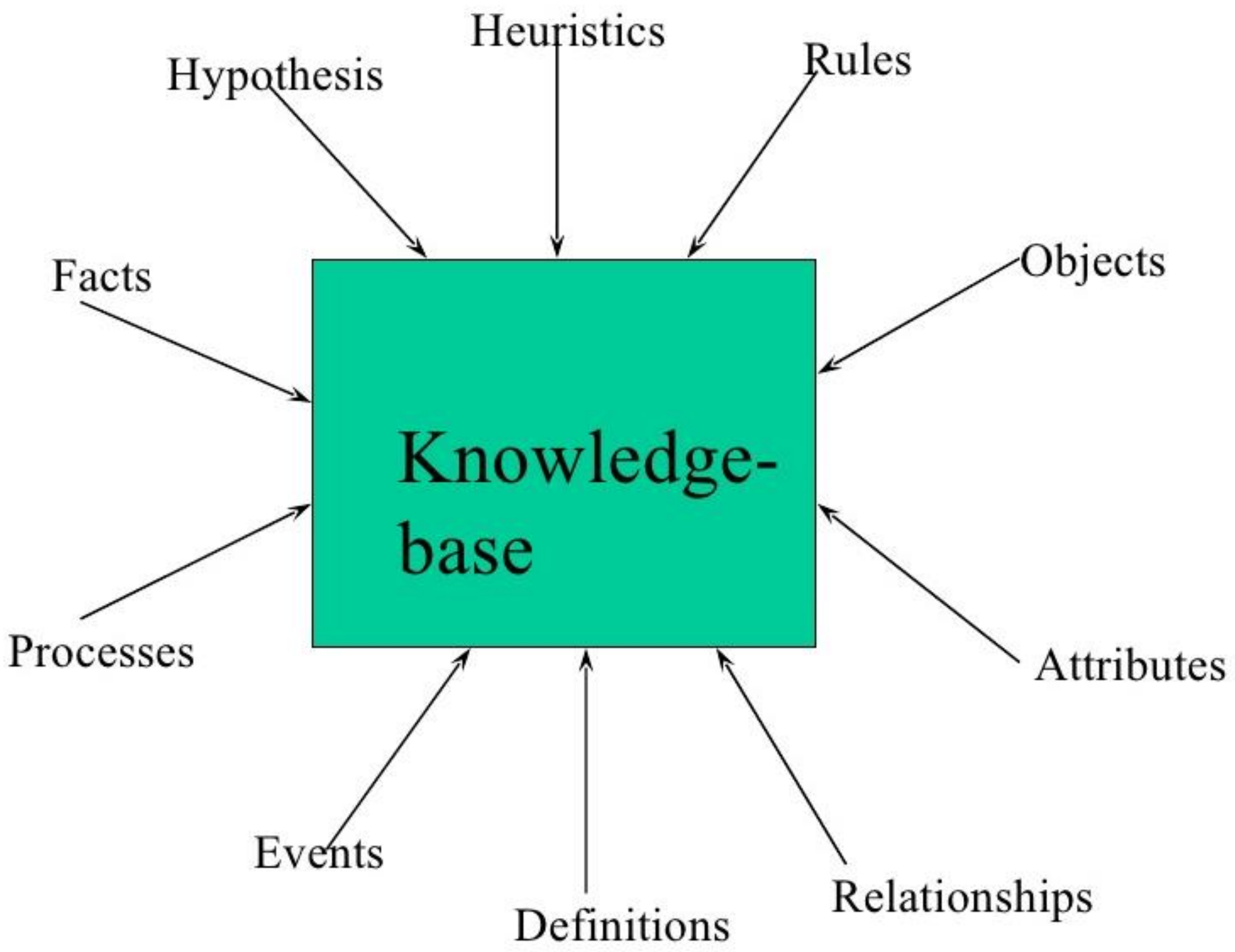
KBS - INFERENCE ENGINE

- It is the component of the system that applies logical rules to the knowledge base to deduce new information
- Two modes:
  1. Forward chaining: starts with the known facts and asserts new facts
  2. Backward chaining: starts with goals and works backward to determine what facts must be asserted so that the goals can be achieved.
- Example of Forward Chaining,
  - B: The road is wet
  - A=>B If it is raining, the road is wet
  - A: It is raining
- Example of Backward Chaining,
  - A: It is raining
  - A=>B If it is raining, the road is wet
  - B: The road is wet

KBS

- Heuristic rather than algorithmic
- Highly specific domain Knowledge
- Knowledge is separated from how it is used
- KBS = knowledge-base + inference engine

(1) Knowledge-base



KNOWLEDGE REPRESENTATION(KR)

KR	Inference
* Logic	→ Resolution principle
* Production Rules	→ backward (top-down, goal directed)
	→ forward (bottom-up, data-driven)
* Semantic nets & Frames	→ Inheritance & advanced reasoning
* Case-based Reasoning	→ Similarity based

KBS

- All forms of reasoning require certain amount of searching and matching.
- Searching and matching both consume the maximum time of computation in AI systems.
- So, we need best searching techniques so that we can avoid this combinatorial explosion problem during searching

HOW TO ACQUIRE KNOWLEDGE?

- Knowledge may be acquired from sources like textbooks, references, reports, technical research papers and so on and to be useful, it should be accurate, complete, inconsistent and so on.
- KBS depends on a high quality knowledge for their success.

EXAMPLE OF KBS

MYCIN (medicine)

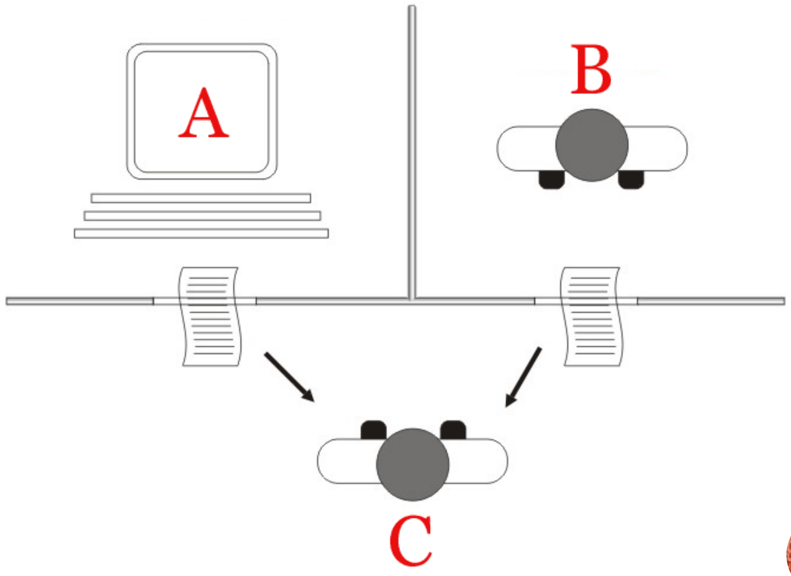
- Assist internists in diagnosis and treatment of infectious diseases
- Given patient data(incomplete & inaccurate) MYCIN gives interim indication of organisms that are most likely causes of infection & drugs to control disease.

CRITERIA FOR SUCCESS

- "How will know if we have succeeded?"
- How will we know if we have constructed a machine that is intelligent?
- Can we do anything to measure our progress?

TURING TEST IN ARTIFICIAL INTELLIGENCE

- The **Turing test** developed by Alan Turing(Computer scientist) in 1950.
- "Turing test is used to determine whether or not computer(machine) can think intelligently like human"
- game of three players
  1. two humans
     1. an interrogator(as human)
     2. Human player
  2. one computer



"STANDARD INTERPRETATION" OF THE TURING TEST

- **Player C,** the interrogator, is given the task of trying to determine which player – A or B – is a computer and which is a human.
- The interrogator is limited to using the responses to written questions to make the determination
- **C(Interrogator):** Are you a computer?
  A(Computer): No
- **C:** Multiply one large number to another, 158745887 * 56755647
  A: After a long pause, an incorrect answer!
- **C:** Add 5478012, 4563145
  A: (Pause about 20 second and then give as answer)10041157

GOAL OF TURING TEST

- The goal of the machine is to fool the interrogator into believing that it is person.
- If machine succeeds at this, then we will conclude that the machine can think.
- If interrogator wouldn’t be able to distinguish the answers provided by both human and computer.
- The computer passes the test and machine(computer) is considered as intelligent as human.
- A computer would be considered intelligent if it’s conversation couldn’t be easily distinguished from a human’s.

SIGNIFICANCE OF TURING TEST

- According to French (1990) when Turing came up with his test, he appeared to be making two claims:
  - The Philosophical Claim: If the machine could pass the test it acts as sufficiently intelligent, therefore it is.
  - The Pragmatic Claim: In the future it would be possible to create a machine that could pass the test.

LIMITATION OF TURING TEST

- The whole conversation would be limited to a text-only channel such as a computer keyboard and screen.

CHINESE ROOM ARGUMENT

- In year 1980, Mr. John searle proposed the "**Chinese room argument**".
- He argued that Turing test could not be used to determine "whether or not a machine is considered as intelligent like humans".
- He argued that any machine like ELIZA and PARRY could easily pass Turing Test simply by manipulating symbols of which they had no understanding.
- Without **understanding**, they could not be described as "thinking" in the same sense people do.

PROBLEM SOLVING

- GPS (General Problem Solver) focus on systems with general capability for solving different types of problems
- Problems represented in terms of
  - Initial state
  - Final state (goal state)
  - A set of legal transitions to transfer states into new states
- Using states & operators, GPS generates sequence of transitions that transform initial state into final state

PROBLEMS WITH GPS

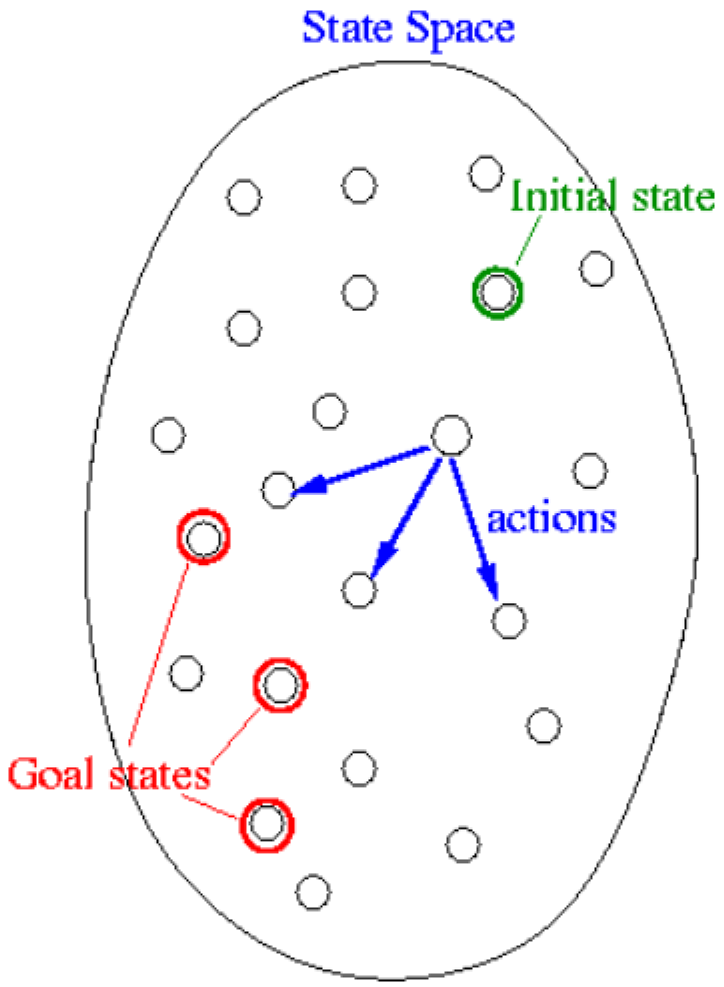
- Efficiency in choosing path to reach the goal
- GPS did not use specific info about problem at hand in selection of state transition
- GPS examined all states leading to exponential time complexity

STATE SPACE SEARCH NOTATIONS

- An **initial state** is the description of the starting configuration of the agent
- An **action** or an operator takes the agent from one state to another state which is called a successor state. A state can have a number of successor states.
- A **plan** is a sequence of actions. The cost of a plan is referred to as the path cost.
- The **path cost** is a positive number, and a common path cost may be the sum of the costs of the steps in the path.
- *Now let us look at the concept of a search problem.*
- Problem formulation means choosing a relevant **set of states** to consider, and a feasible set of operators for moving from one state to another.
- **Search** is the process of considering various possible sequences of operators applied to the initial state, and finding out a sequence which culminates in a goal state.

STATE SPACE SEARCH PROBLEM

- A search problem consists of the following:
  • S: the full set of states
  • s0 : the initial state
  • A:S→S is a set of actions or operators
  • G is the set of final states. Note that G ⊆S

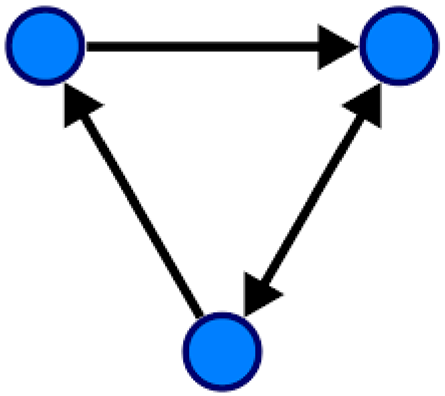


STATE SPACE SEARCH PROBLEM

- The search problem is to find a sequence of actions which transforms the agent from the initial state to a goal state g∈G. A search problem is represented by a 4-tuple {S, s**0**, A, G}.
  S: set of states
  s0 ∈ S : initial state
  A: SÆ S operators/ actions that transform one state to another state
  G : goal, a set of states. G ⊆ S
- This sequence of actions is called a solution plan. It is a path from the initial state to a goal state. A *plan* P is a sequence of actions.
  P = {a**0**, a**1**, … , a**N**} which leads to traversing a number of states {s**0**, s**1**, … , s**N+1**∈G}.
- A sequence of states is called a path. The cost of a path is a positive number. In many cases the path cost is computed by taking the sum of the costs of each action.

REPRESENTATION OF SEARCH PROBLEMS

- A search problem is represented using a directed graph.
  - The states are represented as nodes.
  - The allowed actions are represented as arcs.



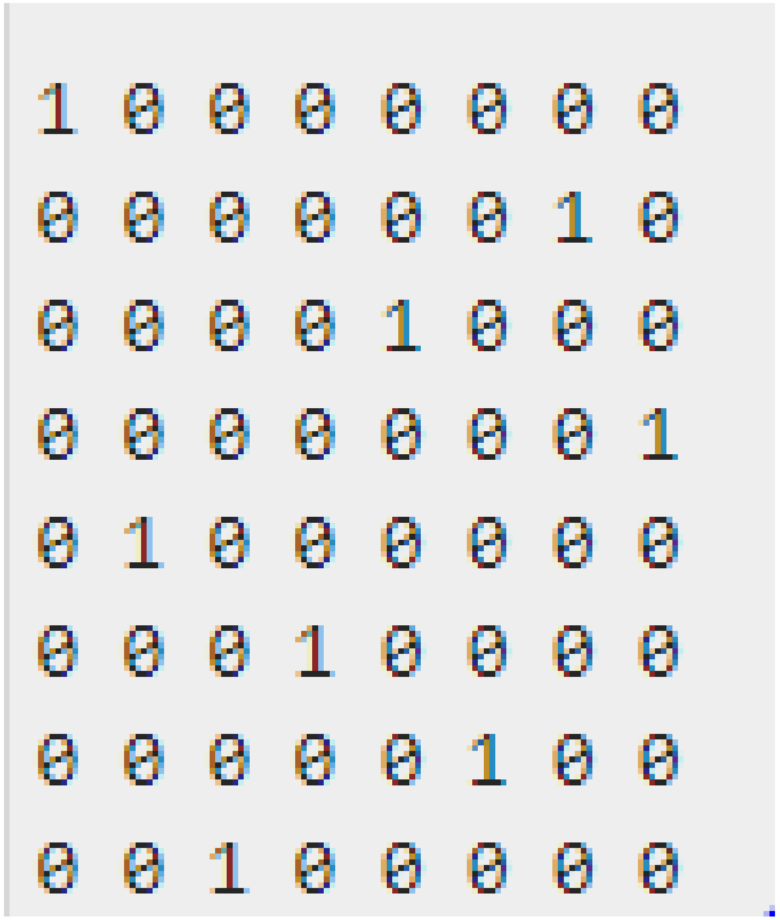
SEARCHING PROCESS

- Do until a solution is found or the state space is exhausted.
- 1. Check the current state
- 2. Execute allowable actions to find the successor states.
- 3. Pick one of the new states.
- 4. Check if the new state is a solution state
- If it is not, the new state becomes the current state and the process is repeated

N QUEENS PROBLEM FORMULATION

- N queens problem formulation 1
  - **States:** Any arrangement of 0 to 8 queens on the board
  - **Initial state:** 0 queens on the board
  - **Successor function:** Add a queen in any square
  - **Goal test:** 8 queens on the board, none are attacked
- N queens problem formulation 2
  - States: Any arrangement of 8 queens on the board
  - Initial state: All queens are at column 1
  - Successor function: Change the position of any one queen
  - Goal test: 8 queens on the board, none are attacked
- N queens problem formulation 3
  - States: Any arrangement of k queens in the first k rows such that none are attacked
  - Initial state: 0 queens on the board
  - Successor function: Add a queen to the (k+1)th row so that none are attacked.
  - Goal test : 8 queens on the board, none are attacked

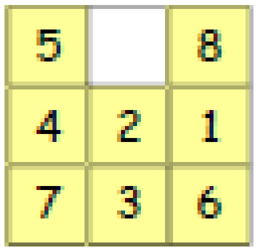
8 QUEENS PROBLEM



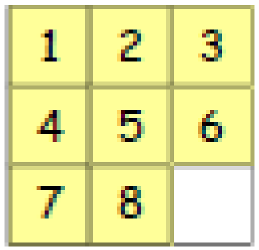
```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

8 PUZZLE PROBLEM

- In the 8-puzzle problem we have a 3×3 square board and 8 numbered tiles.
- The board has one blank position. Blocks can be slid to adjacent blank positions. We can alternatively and equivalently look upon this as the movement of the blank position up, down, left or right.
- The objective of this puzzle is to move the tiles starting from an initial position and arrive at a given goal configuration.
- The 15-puzzle problems is similar to the 8-puzzle. It has a 4×4 square board and 15 numbered tiles.



STATE(N)

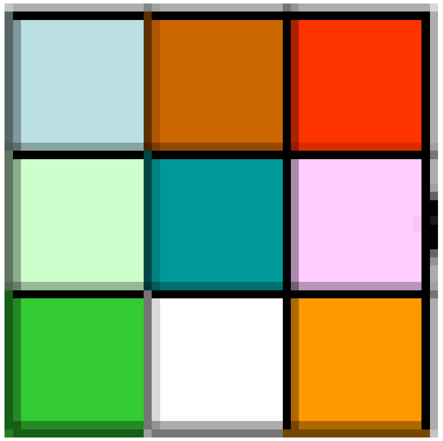


Goal state

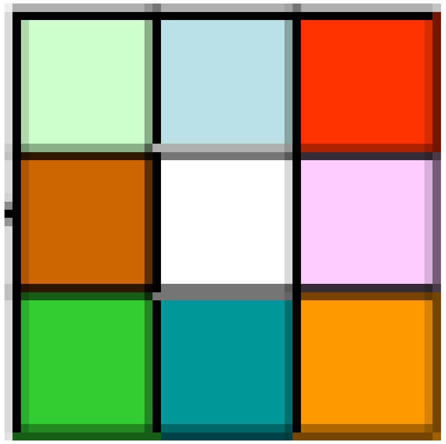
STATE SPACE REPRESENTATION FOR 8-PUZZLE PROBLEM

- **States:** A state is a description of each of the eight tiles in each location that it can occupy.
- **Operators/Action:** The blank moves left, right, up or down
- **Goal Test:** The current state matches a certain state
- **Path Cost:** Each move of the blank costs 1

8-PUZZLE PARTIAL STATE SPACE



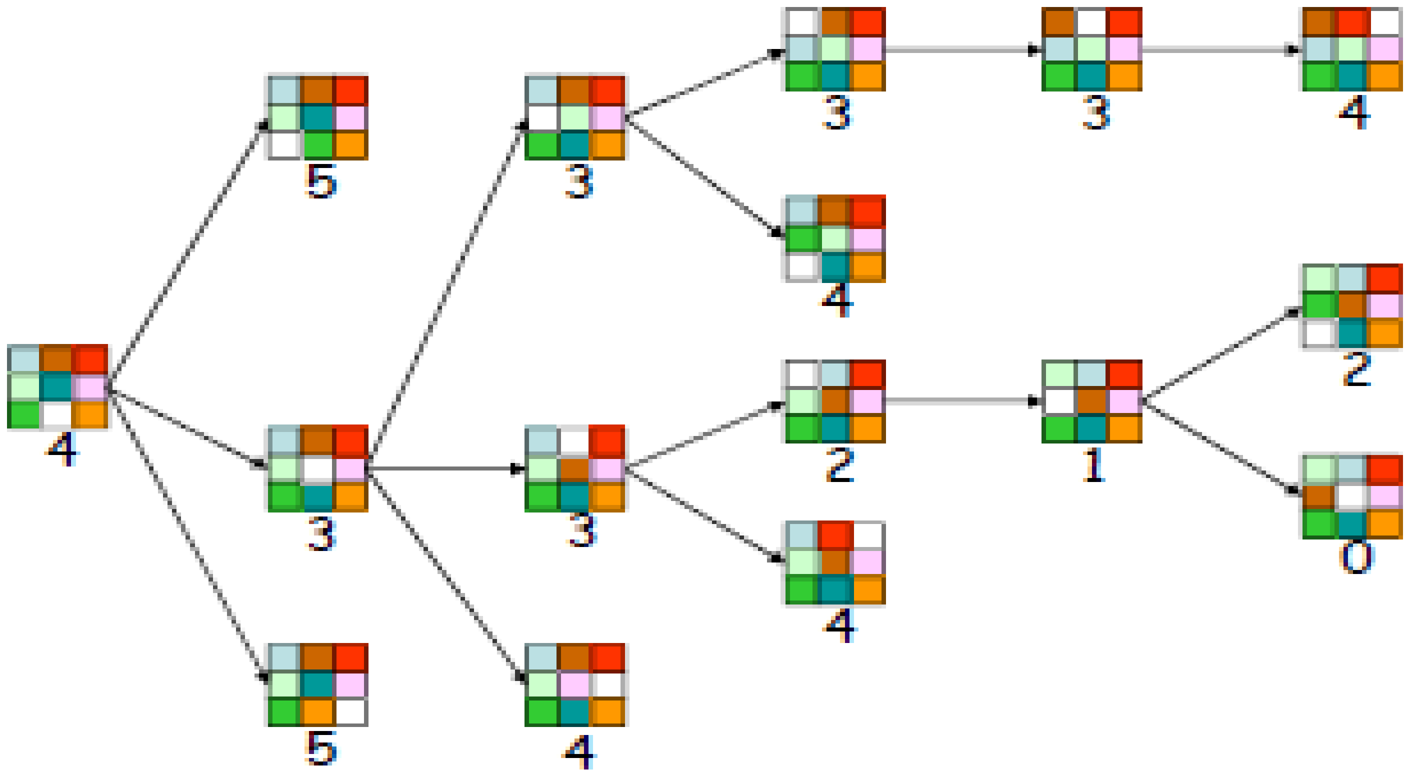
Initial State(N)



Goal State

8-PUZZLE SOLUTION

f(N) = h(N) = number of misplaced numbered tiles

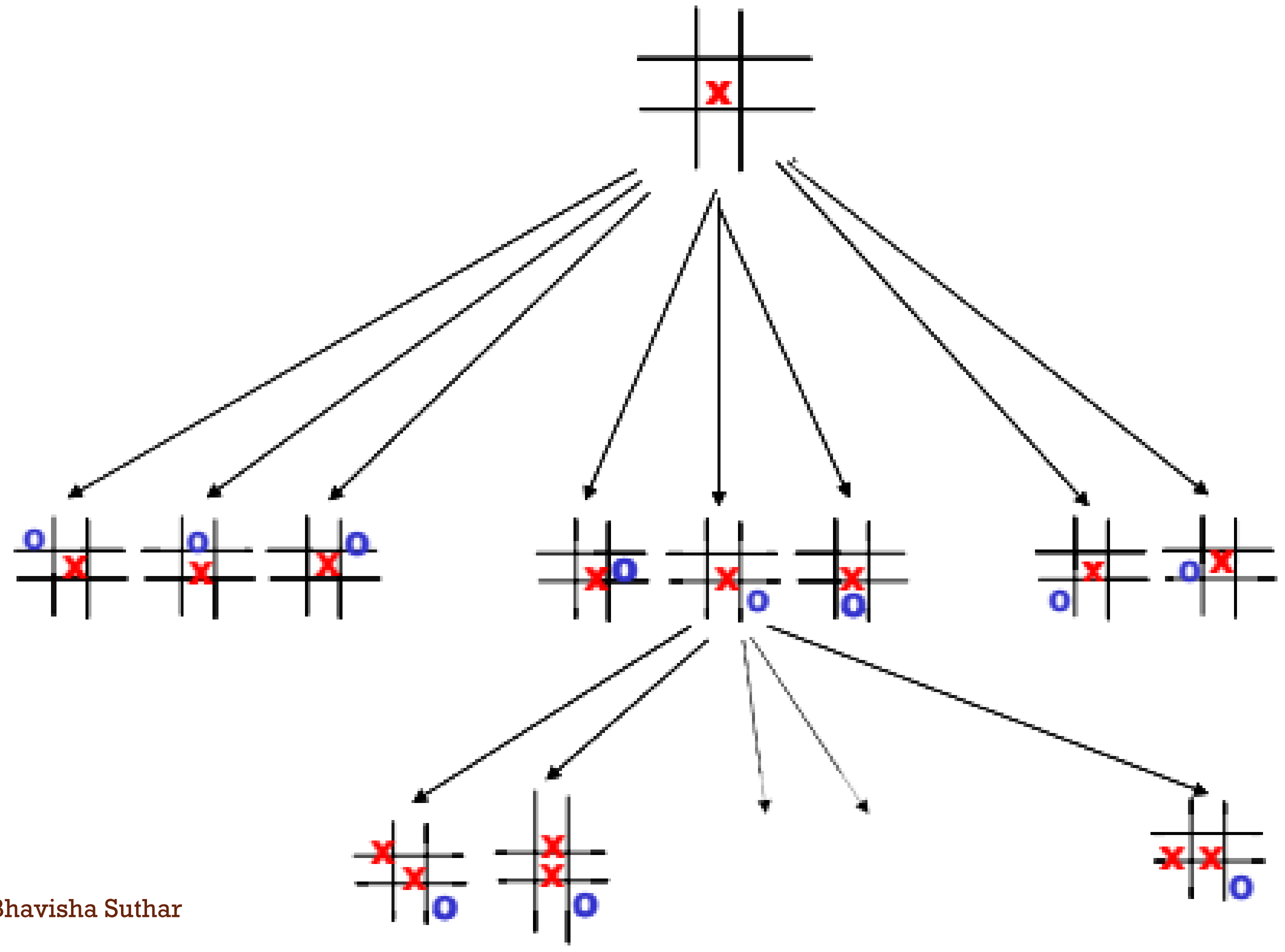


The white tile is the empty tile

PROBLEM DEFINITION - TIC-TAC-TOE

- Problem Definition - Example, tic-tac-toe
  This is a game that involves two players who play alternately.
  Player one puts a X in an empty position.
  Player 2 places an O in an unoccupied position. The player who can first get three of his symbols in the same row, column or diagonal wins.

PORTION OF THE STATE SPACE OF TIC-TAC-TOE



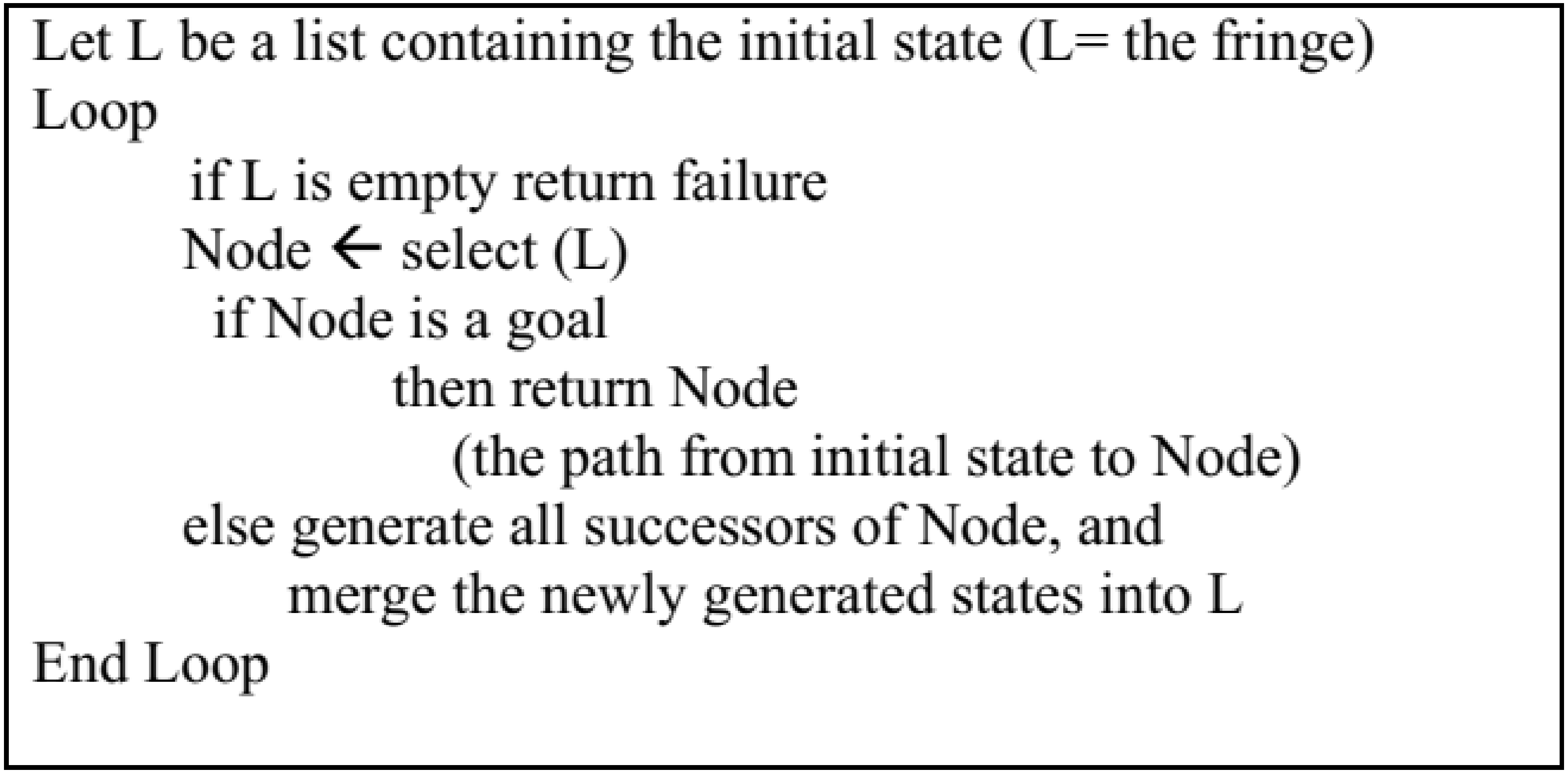
WATER JUG PROBLEM FORMULATION

- You have three jugs measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet.
  You need to measure out exactly one gallon.
  • **Initial state:** All three jugs are empty
  • **Goal test:** Some jug contains exactly one gallon.
  • **Successor function:**
  - Applying the **action** *tranfer* to jugs i and j with capacities Ci and Cj and containing Gi and Gj gallons of water, respectively, leaves jug i with max(0, Gi-(Cj -Gj)) gallons of water and jug j with min(Cj, Gi+ Gj) gallons of water.
  - Applying the **action** *fill* to jug i leaves it with Ci gallons of water.
- **Cost function:** Charge one point for each gallon of water transferred and each gallon of water filled

SEARCH

- Searching through a state space involves the following:
  - A set of states
  - Operators and their costs
  - Start state
  - A test to check for goal state

BASIC SEARCH ALGORITHM



```
Let L be a list containing the initial state (L= the fringe)
Loop
        if L is empty return failure
        Node ← select (L)
         if Node is a goal
                  then return Node
                     (the path from initial state to Node)
        else generate all successors of Node, and
             merge the newly generated states into L
End Loop
```

EVALUATING SEARCH STRATEGIES

- Three factors to measure this:
- 1. **Completeness:** Is the strategy guaranteed to find a solution if one exists?
- 2. **Optimality:** Does the solution have low cost or the minimal cost?
- 3. What is the search cost associated with the time and memory required to find a solution?
  - a. **Time complexity**: Time taken (number of nodes expanded) (worst or average case) to find a solution.
  - b. **Space complexity**: Space used by the algorithm measured in terms of the maximum size of fringe

STATE SPACE SEARCH

- At the core heart of many intelligent processes lies the process of search.
- A search procedure is a strategy for selecting the order in which nodes are generated and a given path selected.
- Classified in these categories:
  1. Blind or uninformed search
  2. Informed or directed search or heuristic search
  3. Constraint Satisfaction Search
  4. Adversary Search

SEARCH TREE – TERMINOLOGY

- **Root Node:** The node from which the search starts.
- **Leaf Node:** A node in the search tree having no children.
- **Ancestor/Descendant:** X is an ancestor of Y is either X is Y's parent or X is an ancestor of the parent of Y. If S is an ancestor of Y, Y is said to be a descendant of X.
- **Branching factor:** the maximum number of children of a non-leaf node in the search tree
- **Path:** A path in the search tree is a complete path if it begins with the start node and ends with a goal node. Otherwise it is a partial path.

NODE DATA STRUCTURE

- A node used in the search algorithm is a data structure which contains the following:
  1. A state description
  2. A pointer to the parent of the node
  3. Depth of the node
  4. The operator that generated this node
  5. Cost of this path (sum of operator costs) from the start state

SEARCH – BLIND SEARCH

- The Blind search algorithms following:-
- Breadth First Search (BFS)
- Depth First Search (DFS)
- Bidirectional Search

BREADTH-FIRST SEARCH - ALGORITHM

1. Put the start node on a list, called OPEN, of unexpanded nodes
2. If OPEN is empty, no solution exists
3. Remove the first node, n, from OPEN and put it in a list, called CLOSED, of expanded nodes
4. Expand node n. If it has no successors, go to (2)
5. Place all successors of n at the end of the OPEN list
6. If any of the successors of n is a goal node, a solution has been found. Otherwise go to (2)

BFS-ALGORITHM FOR AI

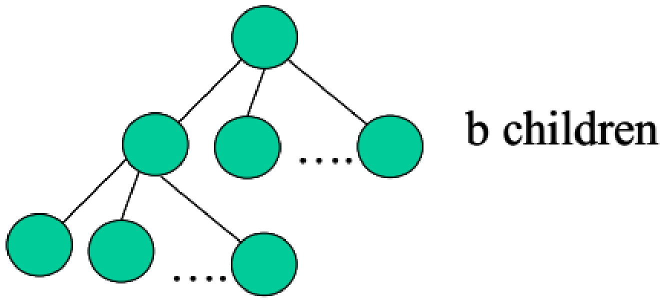
- 1. Create a variable called NODE-LIST and set it to the initial state.
- 2. Until a goal state is found or NODE-LIST is empty:
  - (a) Remove the first element from NODE-LIST and call it E. If NODE-LIST was empty, quit.
  - (b) For each way that each rule can match the state described in E do:
    - (i) Apply the rule to generate a new state.
    - (ii) If the new state is a goal state, quit and return this state.
    - (iii) Otherwise, add the new state to the end of NODE-LIST.

BREADTH FIRST SEARCH

- Breadth first search is:
  • Complete.
  • The algorithm is optimal (i.e., admissible) if all operators have the same cost. Otherwise, breadth first search finds a solution with the shortest path length.
  • The algorithm has exponential time and space complexity.

- Suppose the search tree can be modeled as a b-ary tree as shown in Figure. Then the time and space complexity of the algorithm is O(bd) where d is the depth of the solution and b is the branching factor (i.e., number of children) at each node.

- A complete search tree of depth d where each non-leaf node has b children, has a total of
  $\boldsymbol{1 + b + b^2 + \ldots + b^d = (b^{(d+1)} - 1)/(b-1)}$ **nodes**



BFS-DISADVANTAGE

- Consider a complete search tree of depth 15, where every node at depths 0 to14 has 10 children and every node at depth 15 is a leaf node.
- The complete search tree in this case will have O(1015) nodes. If BFS expands 10000 nodes per second and each node uses 100 bytes of storage, then BFS will take 3500 years to run in the worst case, and it will use 11100 terabytes of memory.
- So you can see that the breadth first search algorithm cannot be effectively used unless the search space is quite small.
- You may also observe that even if you have all the time at your disposal, the search algorithm cannot run because it will run out of memory very soon.

DFS - ALGORITHM

1. Put the start node on a list, called OPEN, of unexpanded nodes
2. If OPEN is empty, no solution exists
3. Remove the first node, n, from OPEN and put it in a list, called CLOSED, of expanded nodes
4. Expand node n. If it has no successors, go to (2)
5. If the depth of node n is greater than the maximum depth, go to (2)
6. Place all successors of n at the beginning of OPEN list
7. If any of the successors of n is a goal node, a solution has been found. Otherwise go to (2)

DFS ALGORITHM FOR AI

- 1. If the initial state is a goal state, quit and return success.
- Otherwise, do the following until success or failure is signaled:
  - (a) Generate a successor, E, of the initial state. If there are no more successors, signal failure.
  - (b) Call Depth-First Search with E as initial state.
  - (c) If success is returned, signal success. Otherwise continue in this loop.

DFS

- The algorithm takes exponential time.
- If N is the maximum depth of a node in the search space, in the worst case the algorithm will take time O(bd).
- However the space taken is linear in the depth of the search tree, O(bN).
- Note that the time taken by the algorithm is related to the maximum depth of the search tree.
- If the search tree has infinite depth, the algorithm may not terminate. This can happen if the search space is infinite.
- It can also happen if the search space contains cycles. The latter case can be handled by checking for cycles in the algorithm. Thus Depth First Search is not complete.

DEPTH LIMITED SEARCH

- A variation of Depth First Search circumvents the above problem by keeping a depth bound.
- Nodes are only expanded if they have depth less than the bound. This algorithm is known as depth-limited search.

SEARCH-HEURISTIC SEARCH

- In blind search the number of nodes can be extremely large
  - The order of expanding the nodes is arbitrary
  - Blind search does not use any properties of the problem being solved
  - Result is the combinatorial explosion
- Information about a particular problem can help to reduce the search
  - The question then is: how to search the given space efficiently
- Heuristic information
  - Additional information beyond that which is built into the state and operator definitions
- Heuristic search
  - A search method using that heuristic information
  - Whether or not the method is foolproof
- Most of the programs were written for a single domain
  - heuristics were closely intertwined in the program and not accessible for study and adaptation to new problems

HEURISTIC SEARCH

- Strategy to limit (drastically) the search for solutions in large problem spaces
- Ways of using heuristic information
  - Which node(s) to expand first instead of expanding is a strictly depth-first or breadth-first manner
  - When expanding a node, decide which successors to generate instead of blindly generate all successors at one time
  - Which nodes not to expand at all (pruning)

CLASSIC AI PROBLEMS

- Traveling Salesman
- Towers of Hanoi
- 8-Puzzle
- Water-Jug Problem
- N-Queen Problem
- Monkey-Banana Problem
- CyptArithmetic Problem
- Missionaries & Cannibals Problem

PRODUCTION SYSTEMS

- A system that uses form of knowledge representation is called a production system.
- A production system consists of rules and factors. Knowledge is encoded in a declarative from which comprises of a set of rules of the form
- Situation → Action
- SITUATION that implies ACTION.
- **Example:-**
- IF the initial state is a goal state THEN quit.

PRODUCTION SYSTEM

- The major components of an AI production system are

i. **One or more knowledge/data bases**

ii. **A set of production rules**

iii. **A control strategy:** specifies order in which the rules will be compared to the database of rules and a way of resolving the conflicts that arise when several rules match simultaneously.

iv. **A rule applier**

CONTROL STRATEGY

Requirements of a good search strategy:

1. It causes motion

   Otherwise, it will never lead to a solution.

2. It is systematic

   Otherwise, it may use more steps than necessary.

3. It is efficient

   Find a good, but not necessarily the best, answer.

FOUR CLASSES OF PRODUCTION SYSTEMS

1. A monotonic production system
2. A non monotonic production system
3. A partially commutative production system
4. A commutative production system.

	Monotonic	NonMonotonic
Partially Commutative	Theorem proving	Robot Navigation
Not Partially Commutative	Chemical Synthesis	Bridge

1. A MONOTONIC PRODUCTION SYSTEM

- In monotonic reasoning, once the conclusion is taken, then it will remain the same even if we add some other information to existing information in our knowledge base. In monotonic reasoning, adding knowledge does not decrease the set of prepositions that can be derived.
- To solve monotonic problems, we can derive the valid conclusion from the available facts only, and it will not be affected by new facts.
- Monotonic reasoning is not useful for the real-time systems, as in real time, facts get changed, so we cannot use monotonic reasoning.
- Monotonic reasoning is used in conventional reasoning systems, and a logic-based system is monotonic.
- Any theorem proving is an example of monotonic reasoning.
- **Example:**
- **Earth revolves around the Sun.**
- It is a true fact, and it cannot be changed even if we add another sentence in knowledge base like, "The moon revolves around the earth" Or "Earth is not round," etc.

2. A NON MONOTONIC PRODUCTION SYSTEM

- In Non-monotonic reasoning, some conclusions may be invalidated if we add some more information to our knowledge base.
- Logic will be said as non-monotonic if some conclusions can be invalidated by adding more knowledge into our knowledge base.
- Non-monotonic reasoning deals with incomplete and uncertain models.
- "Human perceptions for various things in daily life, "is a general example of non-monotonic reasoning.
- **Example:** Let suppose the knowledge base contains the following knowledge:
- **Birds can fly**
- **Penguins cannot fly**
- **Pitty is a bird**
- So from the above sentences, we can conclude that **Pitty can fly**.
- However, if we add one another sentence into knowledge base "**Pitty is a penguin**", which concludes "**Pitty cannot fly**", so it invalidates the above conclusion.

3. A PARTIALLY COMMUTATIVE PRODUCTION SYSTEM

- It's a type of production system in which the application of a sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state x into state Y.
- Theorem proving falls under the monotonic partially communicative system.
- Also a Blocks World Problem.

4. A COMMUTATIVE PRODUCTION SYSTEM.

- These systems are important from an implementation standpoint because they can be implemented without the ability to backtrack to previous states when it is discovered that an incorrect path was followed. This production system increases efficiency since it is not necessary to keep track of the changes made in the search process.

WATER-JUG PROBLEM

- This problem is defined as: " We are given two water jugs having no measuring marks on these. The capacities of jugs are 3 liter and 4 liter. It is required to fill the bigger jug with exactly 2 liter of water. The water can be filled in a jug from a tap."
- In this problem, the start state is that both jugs are empty and the final state is that 4 liter jug has exactly 2 liter of water.
- The production rules involve filling a jug with some amount of water, filing the water from one jug to other or emptying the jug.
- The search will be finding the sequence of production rules which transform the initial state to final state.

WATER-JUG PROBLEM

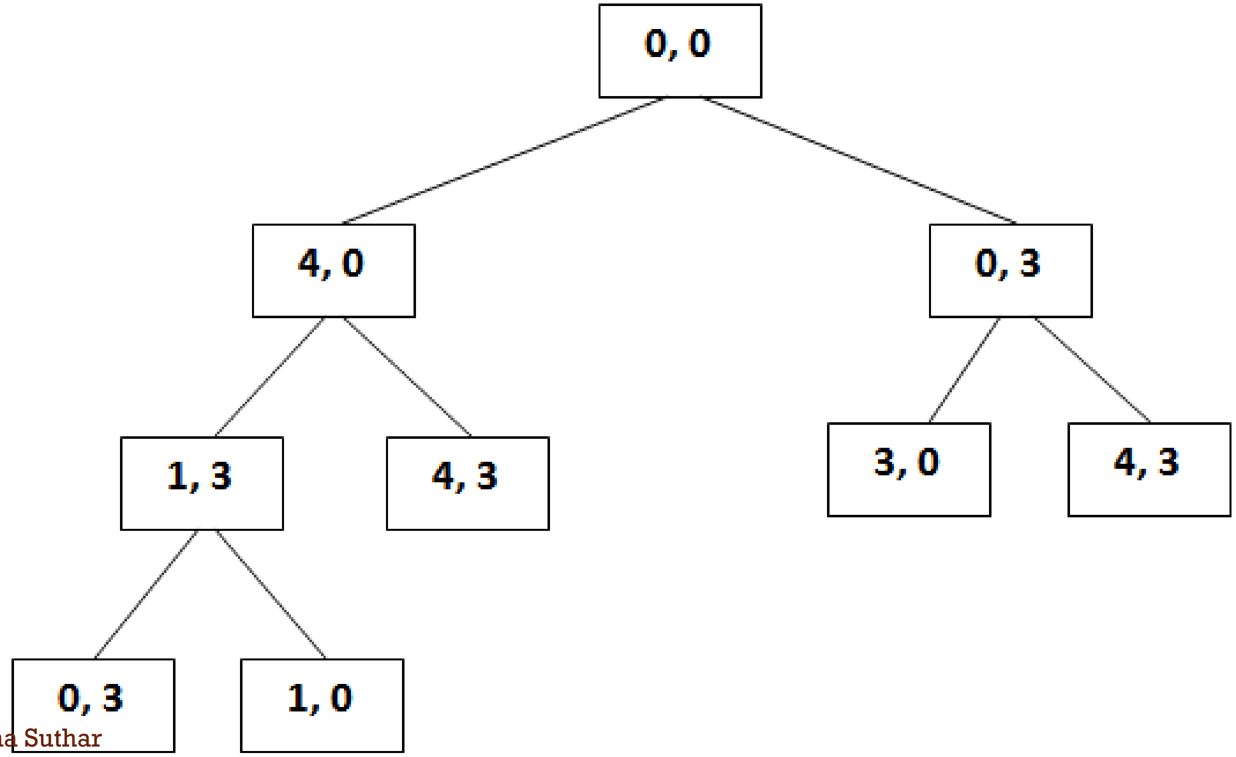
- The production rules are formulated as follows:-
- Rule 1: (x, y) if x < 4 -> (4, y) (Fill the 4 liter jug)
- Rule 2: (x, y) if y < 3 -> (x, 3) (Fill the 3 liter jug)
- Rule 3: (x, y) if x > 0 -> (x-d, y) (Pour some water out from 4 liter jug)
- Rule 4: (x, y) if y > 0 -> (x, y-d) (Pour some water out from 3 liter jug)
- Rule 5: (x, y) if x > 0 -> (0, y) (Empty the 4 liter jug)
- Rule 6: (x, y) if y > 0 -> (x, 0) (Empty the 3 liter jug)
- Rule 7: (x, y) if x + y>= 4 and y > 0 -> (4, y-(4-x)) (Fill the 4 liter jug by pouring some water from 3 liter jug)
- Rule 8: (x, y) if x + y>= 3 and x > 0 -> (x-(3-y), 3) (Fill the 3 liter jug by pouring some water from 4 liter jug)
- Rule 9: (x, y) if x + y <= 4 and y > 0 -> (x + y, 0) (Empty 3 liter jug by pouring all its water in to 4 liter jug)
- Rule 10: (x, y) if x + y <= 3 and x > 0 -> (0, x +y) (Empty 4 liter jug by pouring all its water in to 3 liter jug)

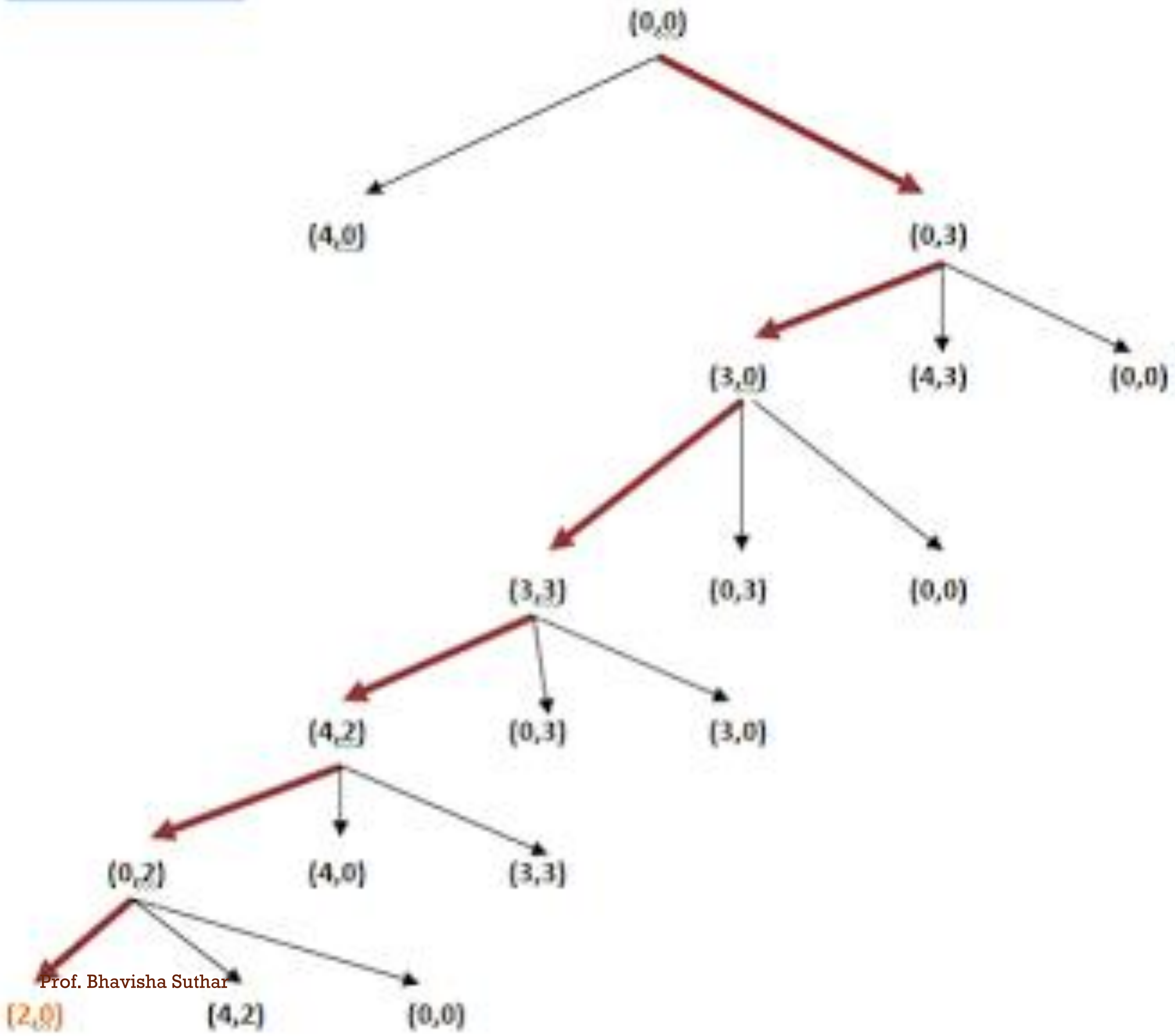
WATER-JUG PROBLEM

Rule Applied	Water in 4 liter jug	Water in 3 liter jug
Start State	0	0
2	0	3
9	3	0
2	3	3
7	4	2
5	0	2
9	2	0

WATER-JUG PROBLEM

- The start state is (0, 0) and the goal state is (2, 0).
- Partial Search tree of water jug problem




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

OPERATIONALIZATION

- In Water jug Problem we discussed and from these discussion it should be clear that the first step towards design of a program to solve a problem must be the creation of a formal and manipulable description of the problem itself.
- Programs can be written so that can themselves produce such formal descriptions from informal ones.
- This process is called operationalization.

CLASSES OF PROBLEMS

1. Ignorable
   - Example, Theorem Proving
2. Recoverable
   - Example, 8-Puzzle
3. Irrecoverable
   - Example, chess

PROBLEM CHARACTERISTICS

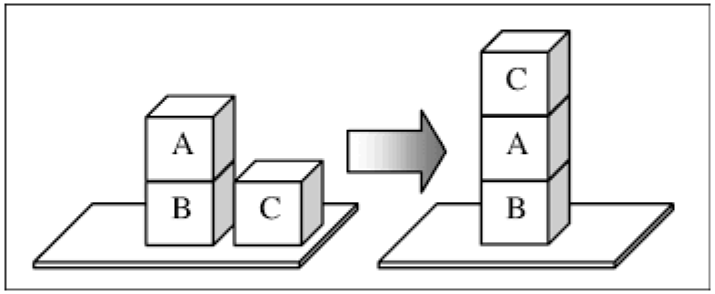
To choose an appropriate method for a particular problem:

- Is the problem decomposable?
- Can solution steps be ignored or undone?
- Is the universe predictable?
- Is a good solution absolute or relative?
- Is the solution a state or a path?
- What is the role of knowledge?
- Does the task require human-interaction?

IS THE PROBLEM DECOMPOSABLE?

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

IS THE PROBLEM DECOMPOSABLE?



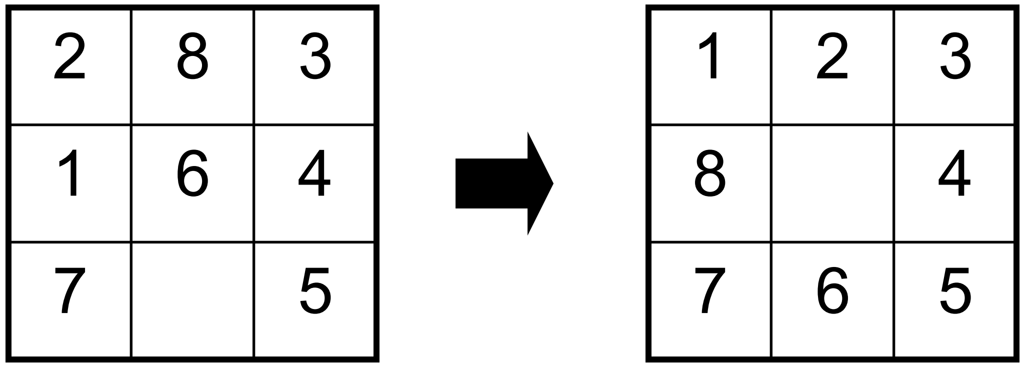
CAN SOLUTION STEPS BE IGNORED OR UNDONE?

Theorem Proving

Ignorable!

CAN SOLUTION STEPS BE IGNORED OR UNDONE?

The 8-Puzzle



Moves can be undone and backtracked.

Recoverable!

CAN SOLUTION STEPS BE IGNORED OR UNDONE?

Playing Chess

Moves cannot be retracted.

Irrecoverable!

CAN SOLUTION STEPS BE IGNORED OR UNDONE?

- Ignorable problems can be solved using a simple control structure that never backtracks.
- Recoverable problems can be solved using backtracking.
- Irrecoverable problems can be solved by recoverable style methods via planning.

IS THE UNIVERSE PREDICTABLE?

The 8-Puzzle

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

Certain outcome!

IS THE UNIVERSE PREDICTABLE?

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. **(eight puzzle and water jug problems)**
- For uncertain-outcome problems, a sequence of generated operators can only have a good probability of leading to a solution. **(playing cards)**

IS A GOOD SOLUTION ABSOLUTE OR RELATIVE?

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 2004 A.D.

IS A GOOD SOLUTION ABSOLUTE OR RELATIVE?

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 2004 A.D.

Is Marcus alive?

IS A GOOD SOLUTION ABSOLUTE OR RELATIVE?

1. Marcus was a man.
2. Marcus was a Pompeian.
3. Marcus was born in 40 A.D.
4. All men are mortal.
5. All Pompeians died when the volcano erupted in 79 A.D.
6. No mortal lives longer than 150 years.
7. It is now 2004 A.D.

Is Marcus alive?

Different reasoning paths lead to the answer. It does not matter which path we follow.

IS A GOOD SOLUTION ABSOLUTE OR RELATIVE?

The Travelling Salesman Problem

We have to try all paths to find the shortest one.

IS A GOOD SOLUTION ABSOLUTE OR RELATIVE?

- Any-path problems can be solved using heuristics that suggest good paths to explore.
- For best-path problems, much more exhaustive search will be performed.

IS THE SOLUTION A STATE OR A PATH?

The Water Jug Problem

The path that leads to the goal must be reported.

IS THE SOLUTION A STATE OR A PATH?

- A path-solution problem can be reformulated as a state-solution problem by describing a state as a partial path to a solution.
- The question is whether that is natural or not.

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.

PROBLEM CLASSIFICATION

- There is a variety of problem-solving methods, but there is no one single way of solving all problems.
- Not all new problems should be considered as totally new. Solutions of similar problems can be exploited.

TRAVELING SALESMAN PROBLEM

- A salesman has a list of cities, each of which he must visit exactly once.
- There are direct roads between each pair of cities on the list. Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities.
- A simple, motion-causing and systematic control structure could, in principle, solve this problem.
- It would simply explore all possible paths in the tree and return the one with shortest length.
- But this approach breaks down quickly as the number of cities grows.

TRAVELING SALESMAN PROBLEM

- If there are N cities, then the number of different paths among them is 1.2…..(N-1) or (N-1)!
- The time to examine a single path is proportional to N. So the total time required to perform this search is proportional to N.
- Assuming there are only 10 cities, 10! = 36,28,800, which is a very large number.
- The salesman have easily have 25 cities to visit. To solve this problem would take more time than he would be willing to spend.
- This phenomenon is called combinatorial explosion.
- It can be solved using *branch-and-bound* technique, but it still requires exponential time.
- It can be solved by Heuristic technique.

QUESTIONS

1. What is Turing's Test? Explain its significance.
2. Define AI. List important task domains of AI.
3. What is AI? What are its application areas?
4. Write different approaches of knowledge representation.
5. In what context did Turing suggest his well known test? Explain the Turing Test.
6. Briefly explain KBS?
7. Describe knowledge acquisition.
8. How does AI solves problems for which no practically feasible algorithm exist?
9. Discuss the difference between declarative and procedural knowledge.
10. Explain the difference between the terms – Knowledge and data.
11. Differences Between Depth First and Breadth First Search.
12. Discuss Production Systems

THANKS!