Artificial Intelligence

Introduction

- How we think, perceive, understand, predict, and manipulate.
- Different types and degrees of intelligence occur in people, several animals, and a few machines.
- AI goes further - it attempts not just to understand but also to build intelligent entities.
- McCarthy coined the term artificial intelligence in 1955, and he created the computer programming language LISP in 1958 which was initially used primarily by the AI community.
- AI currently encompasses a huge variety of subfields.

What is Problem?

- A problem is an obstacle that is troublesome to attain a desired goal, or an issue that is unresolved.
- Each problem has an answer or solution.
- A procedure that makes the problem navigation towards the goal is called solution.
- It can be algorithm or a hard-core implementation to achieve the goal defined by the problem.

Types of Problem and Solution

- Depending on the way in which the problems are solved, problems are classified into the following types:
  1. Structured problem
  2. Unstructured problem

Structured problem

- Structural problems are the ones for which there exists a specific algorithm to achieve the goal.
- The same algorithm is run against variety input data still giving a guarantee of the problem being solved.
- Since the structure of the solution (that is in the algorithm) remains the same, even if the input data changes, these problems are called *structured problems*.
- **For example-**
  - To sort the students according to their total marks, the details about the students may be stored in the database, the sort algorithm will then get the details of students from the database and will solve the problem.

Unstructured problem

- Unstructured problems are the problems for which there does not exist a specific algorithm to achieve the goal.
- What step to take to achieve the goal depends on what is the current state of the problem.
- AI is an attempt to make a computer to solve unstructured problems.
- **For example-**
  - A problem of playing chess or a problem to write a program to perform heart surgery, etc., are unstructured problems, because there does not exist any specific algorithm to solve it. Such problems are solved using a Knowledge Base.

What is AI?

- The study of mental facilities through the use of computational models. - ***Charniak and McDermott, 1985.***
- *The study of how to make computer do things at which at the moment, people are better****. – Rich and Knight 1991***
- The scientific understanding of the mechanisms underlying thought and their intelligent behavior and their embodiment in machines. - ***American Association of Artificial Intelligent (AAAI)***
- AI strives to understand and build intelligent entities (also help us to learn more about ourselves – ***Russell and Norvig.***
- AI is The science and engineering of making intelligent machines, especially intelligent programs. - ***John McCarthy, father of Artificial Intelligence.***

What is AI

- Top - thought processes and reasoning,
- Bottom - behaviour.
- Left - measure success in terms of fidelity to human performance.
- Right - measure against an ideal performance measure, called rationality.
- A system is rational if it does the "right thing," given what it knows.

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)

Acting humanly: The Turing Test approach

- The computer would need to possess the following capabilities:
  - Natural language processing - to communicate successfully
  - 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
  - Computer vision to perceive objects
  - Robotics to manipulate objects and move about.

Thinking humanly: The cognitive approach

- If a program thinks like a human, we must have some way of determining how humans think.
- We need to get inside the actual workings of human minds.
- Cognitive science brings together computer models from AI and experimental techniques from psychology to construct precise and testable theories of the human mind.

Thinking rationally : laws of thought approach

- Greek philosopher Aristotle provided irrefutable reasoning processes called syllogisms.
- Syllogisms provides patterns for argument structures that always yielded correct conclusions when given correct premises - Example.
- The laws of thought were supposed to govern the operation of the mind, their study initiated the field called logic.
- The socalled logicist tradition within artificial intelligence hopes to build on such programs to create intelligent systems.

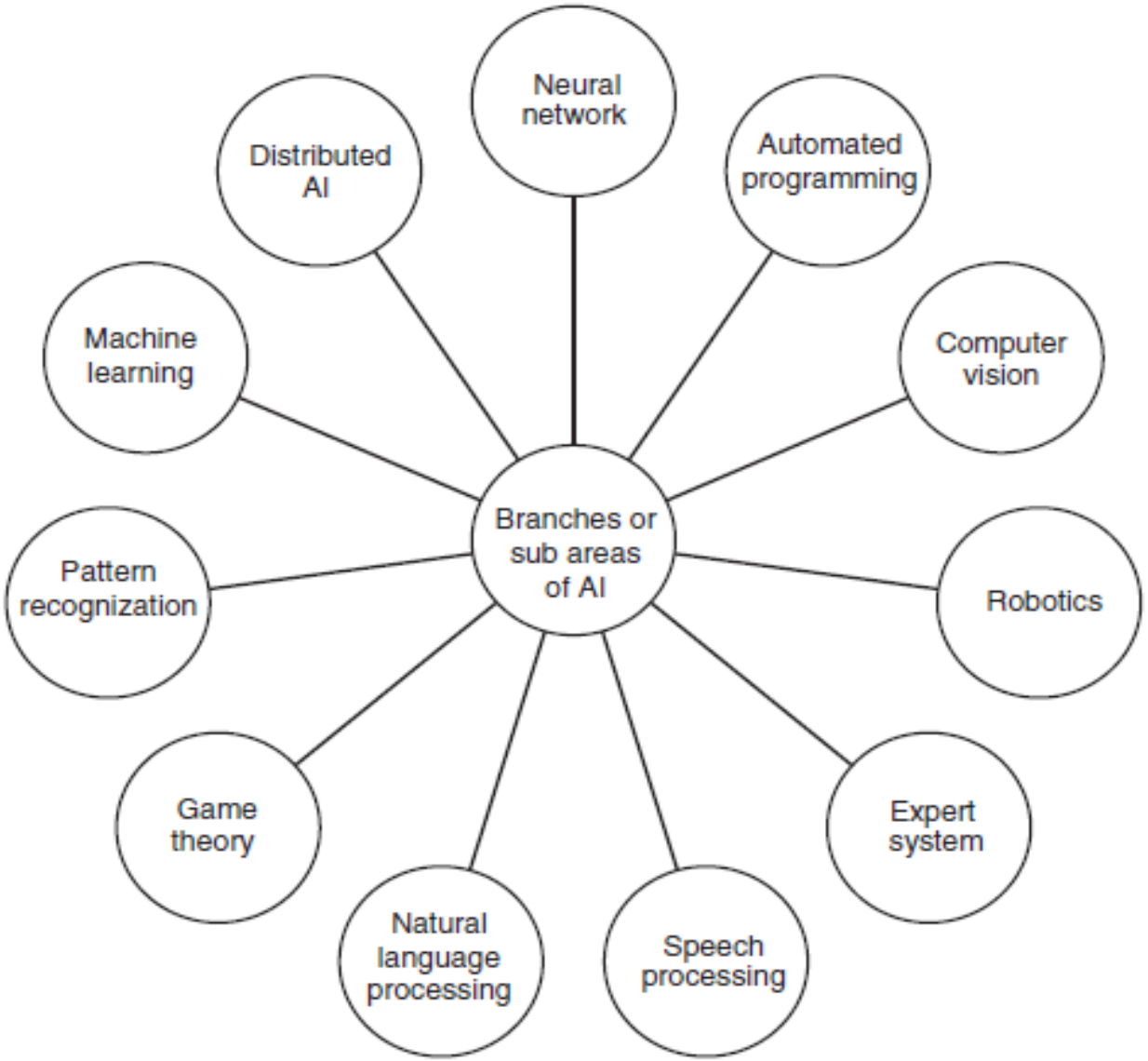
Acting rationally: The rational agent approach

- An agent is just something that acts (agent comes from the Latin agere, to do).
- Computer agents are expected to do more than simple computer programs - operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals.
- A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.
- Limited rationality - acting appropriately when there is not enough time to do all the computations one might like.

History and Foundation of AI

1. 1943–1952 : Beginning (model of neuron , Turing Test, first neural network)
2. 1952–1969: Early enthusiasm (Sobering up, Game of Checkers, Logic Theory, LISP, machine translation, Scaling up to larger problems)
3. 1956 : The birth of artificial intelligence
4. 1970–1979: Knowledge-based systems (DENDRAL : 450 rules, MYCIN : 550 rules, Natural Language Processing, PROLOG – Logical programming language, Frame Theory for data structure to represent a situation)
5. 1980–2010: AI becomes an industry (several million to billion dollars, era of Intelligent Agents, Machine Learning, Robotics.
6. 2010–till date: Era of Deep Learning (stunning advancement, availability of large dataset)

Branches of AI



AI - State of the Art

- ➢Game playing
- ➢Mathematics
- ➢Autonomous control
- ➢Diagnosis
- ➢ Logistics planning
- ➢Autonomous planning and scheduling
- ➢Language understanding and problem solving
- ➢Robotics
- ➢Natural Language Generation

Applications of AI

- Speech recognition
- Virtual agents
- Text analytics and NLP
- Robotic process automation
- Biometrics
- Deep learning platforms
- Decision management
- AI-optimized hardware
- Machine-learning platforms

AI Programming languages

- A number of programming languages exist that are used to build AI systems.
- Here are some languages that are most typically used for creating the AI projects:
  - ➢ PROLOG
  - ➢ LISP
  - ➢ R
  - ➢ Python
  - ➢ Java
  - ➢ C++

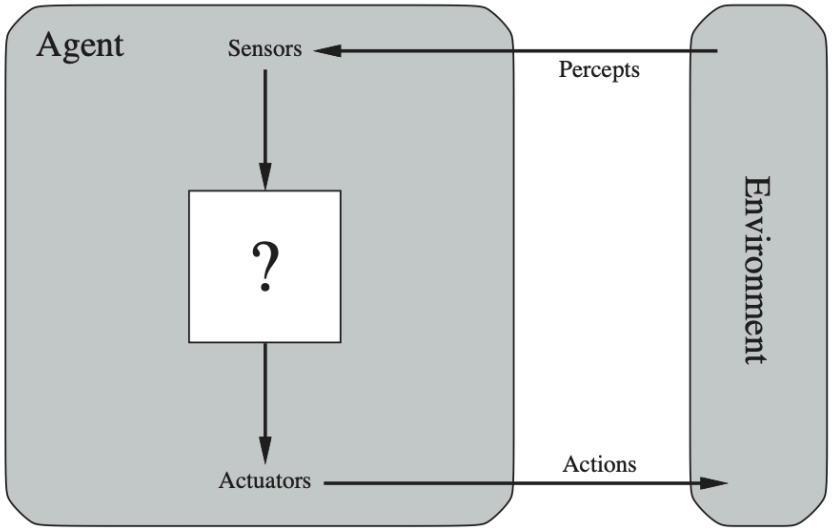
Intelligent Agents

Agents and Environments

- Intelligent Agent (IA) is a self sufficient substance which watches and follows up on a situation and coordinates its movement towards accomplishing objectives.
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- These may be extremely straight forward or extremely complex.
- Small set of design principles for building successful agents systems that can reasonably be called intelligent.

Structure of Intelligent Agent

- The structure of agent can be represented as:
  Agent= Architecture + Program
  - Architecture : The machinery that an agent executes on
  - Agent program: An implementation of an agent function



Agent terminology

1. **Performance measure :** It is the criteria determining the success of an agent.
2. **Action :** It is the action performed by an agent after any specified sequence of the percepts.
3. **Percept :** It is defined as an agent's perceptual inputs at a specified instance.
4. **Percept sequence :** It is defined as the history of everything that an agent has perceived till date.
5. **Agent function (F) :** It is defined as a map from the precept sequence to an action.

Agent function, $a = F(p)$

where *p is the current percept, a is the action carried out, and F is the agent function*

Agent terminology

- If P is the set of all precepts, and A is the set of all actions then F maps precepts to actions.

  $F : P \rightarrow A$
- Generally, an action may be dependent of all the precepts observed, not only the current percept,

  $a_k = F(p_0\, p_1\, p_2 \ldots p_k)$

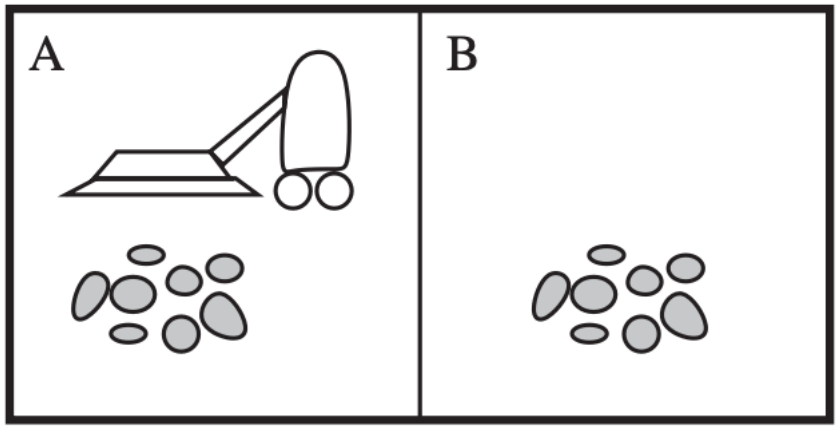
  Where p_0 , p_1, p_2, … ,p_k is the sequence of percepts recorded till date, a_k is the resulting action carried out and F now maps percept sequences to action

  $F : P^* \rightarrow A$

Vacuum cleaner Agent

- The goal of vacuum cleaner is to clean the environment.
- The environment has two locations of the room as square A(left Square) and Square B(Right Square).
- Vacuum cleaner perceives in which square it is in and if there is dirt in the square.
- It may choose to move left, right, suck up the dirt or do nothing.
  - Goal / Performance : All the rooms are well cleaned.
  - Actions : Left, right, suck and no-op (Doing nothing).
  - Percept : Location and status, for example, [A, Dirty].
  - Agent function : Mapping of precept sequence to an action.

Vacuum cleaner Agent



Percept sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck
[A, Dirty], [A, Clean]	Right
[A, Clean], [B, Dirty]	Suck
[B, Dirty], [B, Clean]	Left
[B, Clean], [A, Dirty]	Suck
[A, Clean], [B, Clean]	No-op
[B, Clean], [A, Clean]	No-op

Function REFLEX-VACUUM-AGENT([*location, status*]) **returns**
Action **if** *status* = *Dirty* **then return** *Suck*
if *location* = A **then return**
Right **if** *location* = B **then**
return *Left* **return** action.

Rational Agent

- 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.
- A rational agent : 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.

Agents and Environments

- The irrational agent
- Penalty
- A better agent - do nothing
- information gathering
- Learning
- Autonomy
- A rational autonomous agent

The task environment (PEAS)

- Specify the task environment as fully as possible.
- All these are grouped under the heading of the task environment called PEAS (Performance, Environment, Actuators, Sensors) description.
- **P, Performance :** What the agent is expected to achieve.
- **E, Environment :** What the agent is interacting with
- **A, Actuators :** The outputs of the AI system.
- **S, Sensors :** The inputs to the AI system.

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

The task environment

- More complex problem: an automated taxi driver. The full driving task is extremely open-ended.

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

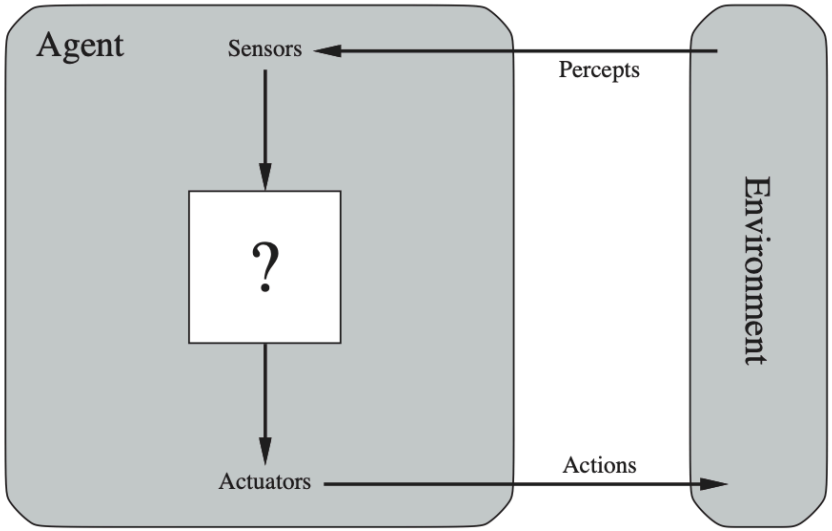
The task environment

- Fully vs. partially observable
- Single agent vs. multiagent
- Deterministic vs. stochastic
- Episodic vs. Sequential environments
- Static vs. dynamic
- Discrete vs. continuous
- Known vs. unknown
- The hardest case is partially observable, multiagent, stochastic, sequential, dynamic, continuous, and unknown.

The task environment

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

THE STRUCTURE OF AGENTS



---

function TABLE-DRIVEN-AGENT(*percept*) **returns** an action
  persistent: *percepts*, a sequence, initially empty
    table, a table of actions, indexed by percept sequences, initially fully specified

  append *percept* to the end of *percepts*
  action ← LOOKUP(*percepts*, *table*)
  return *action*

- The agent program implements the agent function - the mapping from percepts to actions.

THE STRUCTURE OF AGENTS

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

THE STRUCTURE OF AGENTS

- Four types of agent programs that embody the principles underlying almost all intelligent systems:
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agent

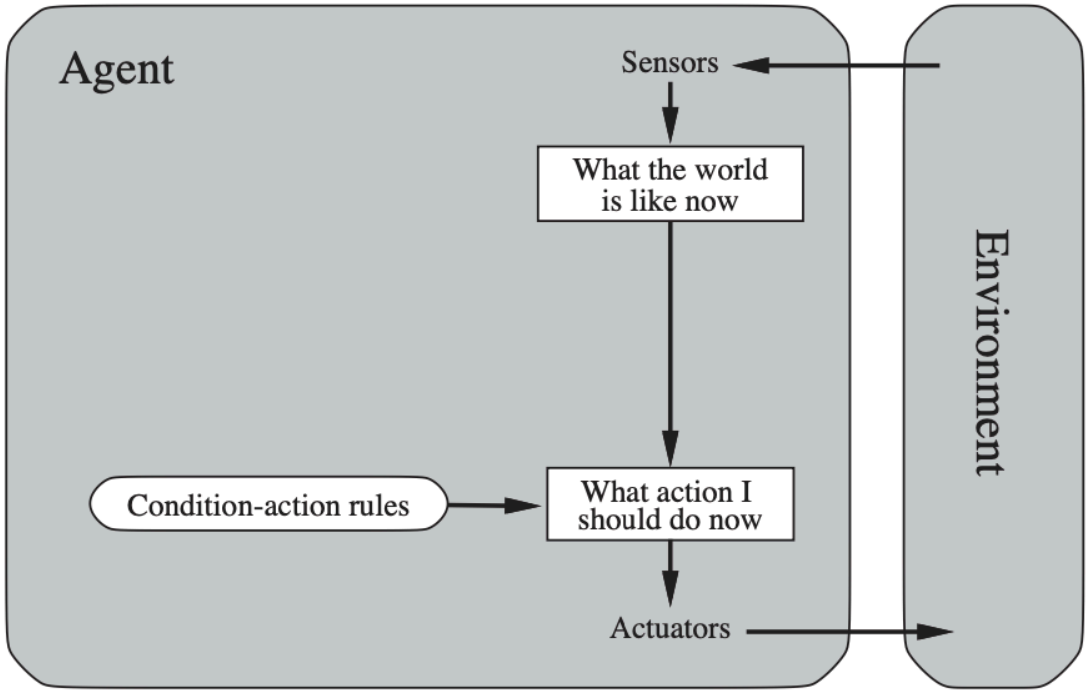
Simple reflex agents

- The simplest kind of agent.
- Select actions on the basis of the current percept, ignoring the rest of the percept history.
- Then triggers some established connection in the agent program to perform the action

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*

Simple Reflex Agents



function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
persistent: *rules*, a set of condition–action rules

state ← INTERPRET-INPUT(*percept*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

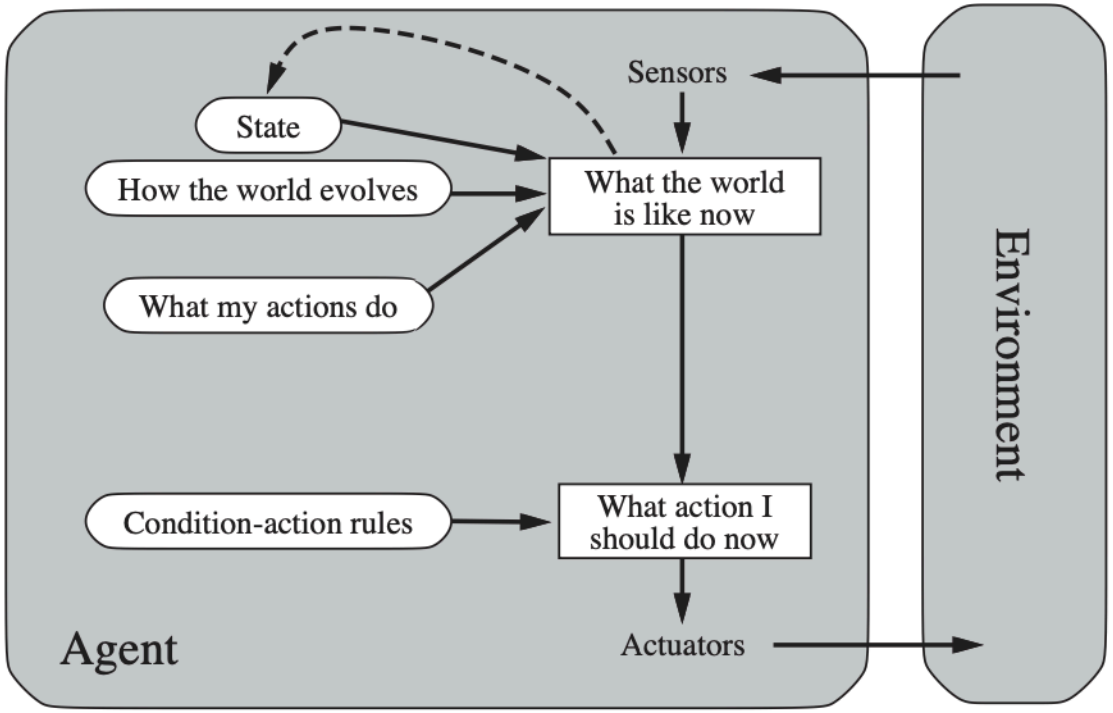
Limitations

- Intelligence level in these agents is very limited.
- It works only in a fully observable environment.
- It does not hold any knowledge or information of non perceptual parts of state.
- Because of the static knowledge based, it is usually too big to generate and store.
- If any change in the environment happens, the collection of the rules are required to be updated.

Model-based reflex agents

- One problem with the simple reflex agents is that their activities are dependent of the recent data provided by their sensors.
- More effective way to handle partial observability is to keep track of the part of the world it can't see now.
- The agent maintains 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.
- An agent that uses such a model is called a model-based agent.
- Model-based agent is known as *Reflex agents with an internal state.*

Model Based Agents



function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action

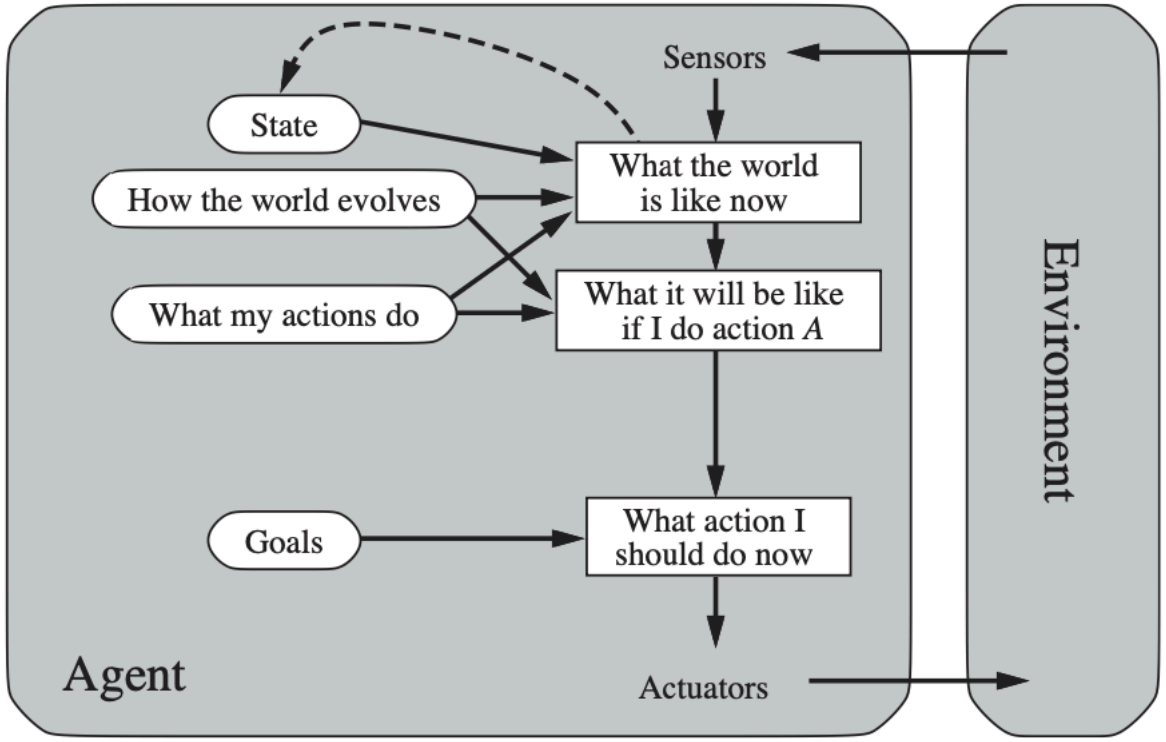
persistent: *state*, the agent's current conception of the world state
model, a description of how the next state depends on current state and action
rules, a set of condition–action rules
action, the most recent action, initially none

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

Goal-based agents

- Knowing something about the current state of the environment is not always enough to decide what to do.
- The agent needs some sort of goal information that describes situations the final destination.
- The agent program can combine this with the model based reflex agent to choose actions that achieve the goal.
- Goal-based action selection may be straightforward or tricky. Searching and planning are the area of study to find action sequence that achieve agents goal.
- They are more adaptable and flexible.
- One can specify another goal rather than reprogramming all the rules.

Goal-based agents



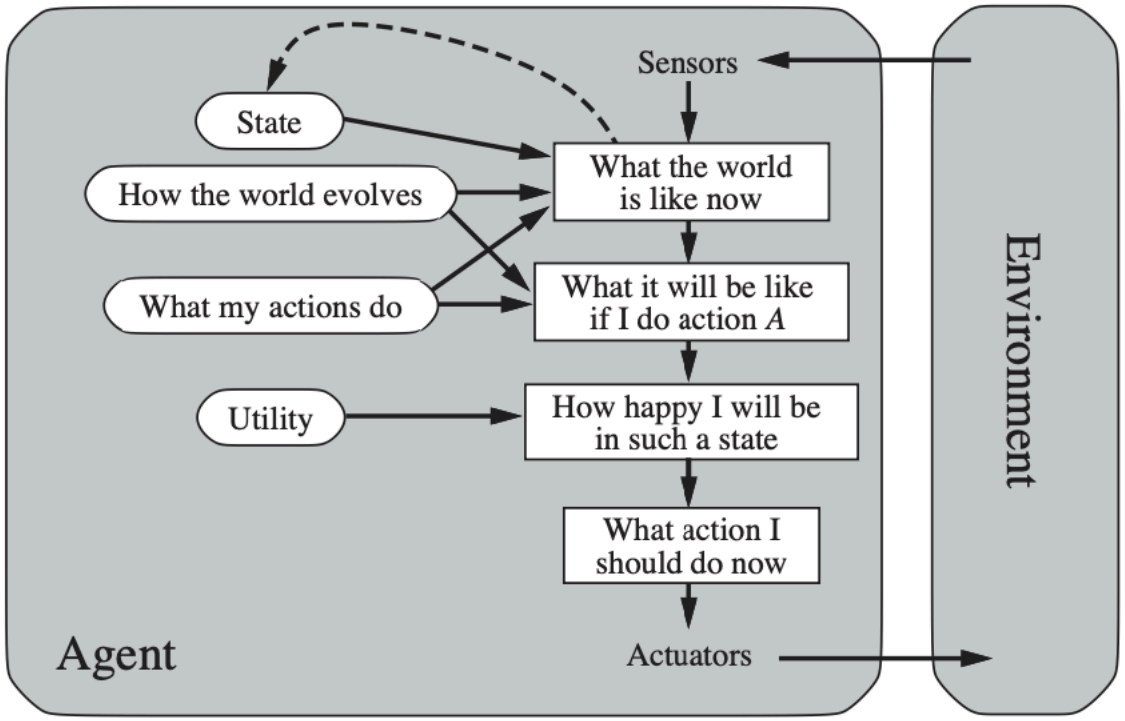
Function REFLEX-AGENT-WITH-GOAL (*percept*) **returns** action.
persistent: *state*, conception of world state.
persistent: *rules*, a set of condition–action rules.
persistent: *action*, the most recent action, initially none.
state = UPDATE-STATE (*state, action, percept*).
rule = MATCH-RULE (*state, rules*).
rule = GOAL-TEST (*rules*)
action=RULE-ACTION (rule)
return *action*

Goal-based agents

- Goal-based agent appears less efficient but it is more flexible due to the knowledge that supports its decisions is represented explicitly and can be modified.
- This will automatically cause all of the relevant behaviors to be altered to suit the new conditions.
- The goal-based agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal.
- For the reflex agent, on the other hand, we would have to rewrite many condition–action rules.
- The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new.

Utility based agent

- Goals individually are insufficient to produce top high-quality behavior.
- When there are conflicting goals, only some of which can be achieved (e.g 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.
- Given proper criteria, it might be conceivable to pick ‘best’ sequence of actions from a number that all result in the goal being achieved.
- Any utility-based agent can be depicted as having an utility capacity that maps a state, or grouping of states, on to a genuine number that speaks to its utility or convenience or usefulness.
- A rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes.

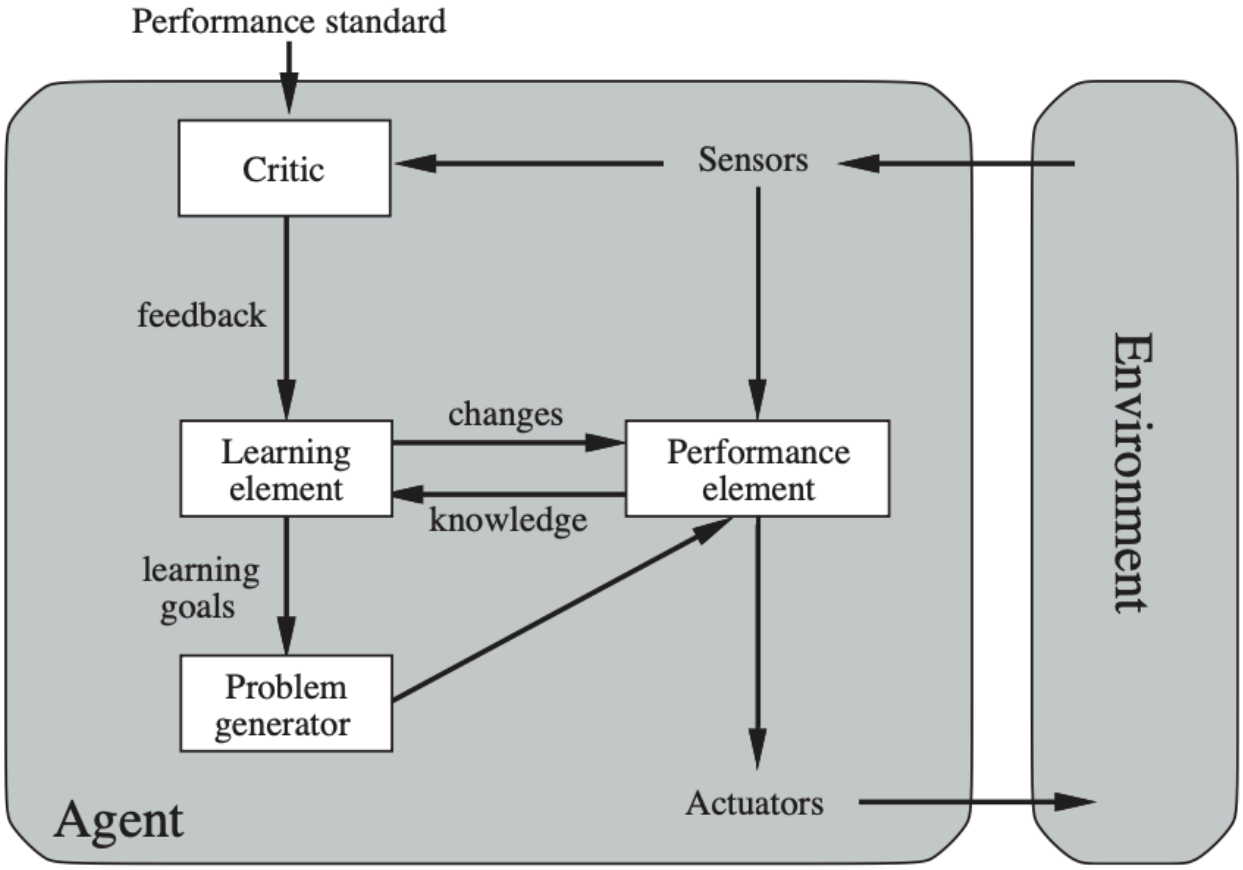


Function REFLEX-AGENT-WITH-UTILITY(*percept*) **returns** action
persistent: *state*, conception of world state
persistent: *rules*, a set of condition-action rules
persistent: *action*, the most recent action, initially none
state = UPDATE-STATE(*state, action, percept*)
rule = MATCH-RULE(*state, rules*)
rule = UTILITY-OF-RULE(*rules*)
action=RULE-ACTION(rule)
return *action*

Learning agent

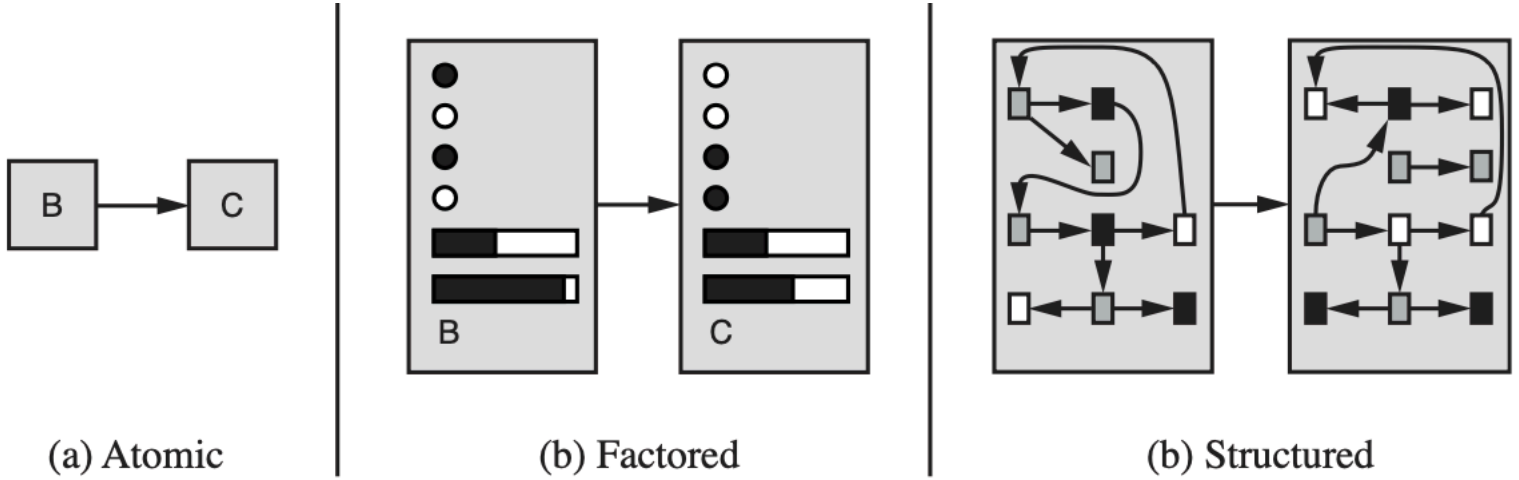
- By actively exploring and experimenting with their environment, the most powerful agents are able to learn.
- A learning agent can be further divided into the four conceptual components
  - performance element - responsible for selecting external actions.
  - learning element - responsible for making improvements
  - problem generator - responsible for suggesting actions that will lead to new and informative experiences
  - Critic - Provides feedback on how agent is doing and determines how the performance element should be modified to do in future.
- The performance element is what we have previously considered to be the entire agent, it takes in percepts and decides on actions.
- The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

Learning agent



Components of Agent Programs

- Representations along an axis of increasing complexity and expressive power - atomic, factored, and structured.



Components of Agent Programs

- Atomic - Each state of the world is indivisible, it has no internal structure.
- Factored - splits up each state into a fixed set of variables or attributes, each of which can have a value.
- Two different atomic states have nothing in common and can share some attributes (GPS location) and not others (fuel status), this makes it much easier to work out how to turn one state into another.
- We would need a structured representation, in which objects and their various and varying relationships can be described explicitly.
- Structured representations underlie relational databases and first-order logic In fact, almost everything that humans express in natural language concerns objects and their relationships.

PROBLEM SOLVING
- Solving problem by searching

Introduction

- In AI, the term *problem solving* is given to the analysis of how computers can be made to find solutions in restricted environments.
- *Problem solving* is defined as the way in which an agent finds a sequence of actions that achieves its goals, when no single action will do.
- *Problem formulation* requires abstracting real-world details to define a state space that can be explored.
- This step of abstraction is performed by an agent called as a *problem-solving agent.*
- Intelligent agents are supported to act in such a way that the environment goes through a sequence of states that maximises the performance measures.

Problem Solving Agent

Problem solving agent is a form of goal based agent. Consider a goal to be a set of world states in which a goal is satisfied.

1. **Goal state**: A state that describes the objective that the agent is trying to achieve is called as the goal state of the agent.
2. **Action**: When there is a transition between the world states an action is said to be performed.

Problem Solving Agent

Steps in problem solving by problem solving agent

1. **Goal formulation**: On the basis of the current situation and agent's performance measure, it is the first step in problem solving. Agent's task is to find out sequence of actions that will get it to the goal state. Goals help organise behaviour by limiting the objectives the agent is trying to achieve.
2. **Problem formulation**: It is a process of deciding what actions and states to consider given a goal.
3. **Choosing the best sequence:** An agent with several immediate options of unknown value can decide what to do by first examining different possible sequences of actions that lead to the states of known value and then choosing the best sequence.

Problem Solving Agent

- ***Problem formulation*** is the process of deciding what actions and states to consider given a goal.
- It is defined by four component
  - *Initial State* : The starting state of the agent.
  - *Successor Function* : Function that gives the description of all possible action available to the agent.
  - *Goal test* : this determines the given state is goal state or not.
  - *Path Cost* : The function that assigns the numeric cost to each path. The problem solving agent chooses a cost function that reflects its own performance measures.

Problem Solving Agent

- A simple problem solving goal based agent 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.

```
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
```

State-space Representation

- State space is a tree which represents all the possible states in the problem.
- It is complete set of states including start and goal states, other valid states and transitions, where the answer of the problem is to be searched.
- A state–space representation allows for the formal definition of a problem, which makes the movement from initial state to the goal state quite easily.
- State–space search is a process in AI, in which successive configurations or states of an instance are considered, with the goal of finding a goal state with a desired property.

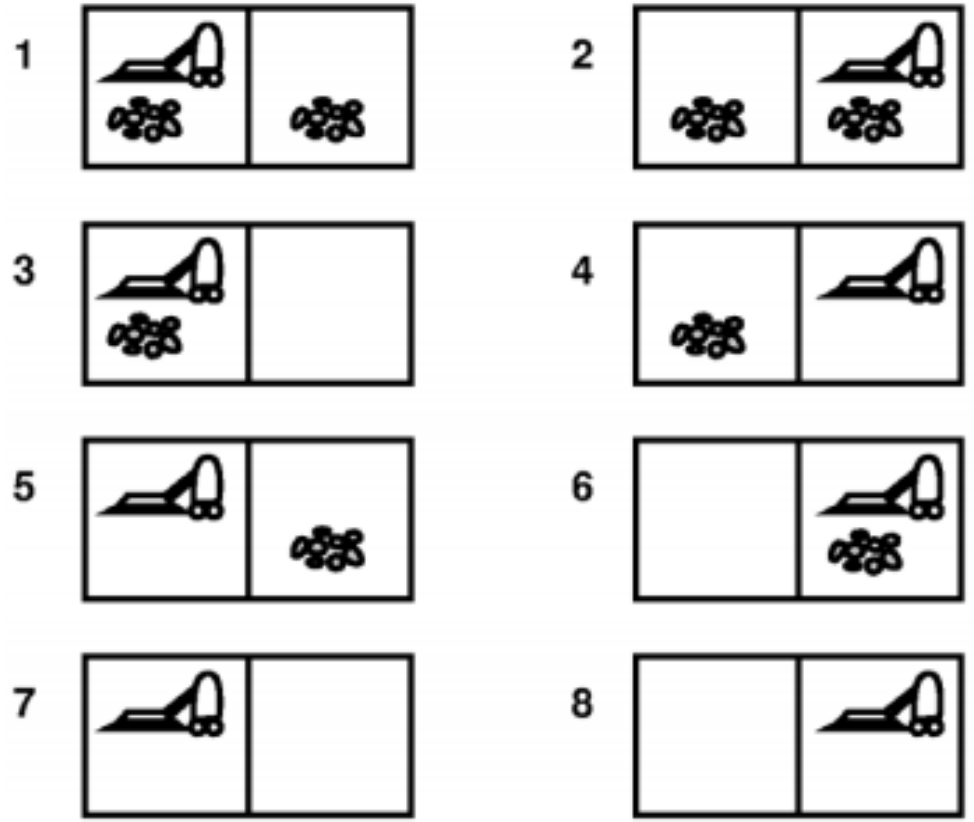
State-space Representation

- AI problem can be solved using following steps:
  - Describe states
  - Identify initial state
  - Identify set of rules (all possible action)
  - Identify goal state
  - Path cost
- When these steps are followed in problem formulation then a problem is said to be formulated by an approach called as state space approach.
- Find Problem solution path in the state space.
- Various problems like planning, learning, theorem proving, etc, are all essentially search problems.

Problem Formulation – Vacuum Cleaner

Problem Definition :

- Consider just two location of vacuum cleaner.
- Each location may or may not contain dirt and agent may be in one location or the other.
- The goal of the agent is to clean 100% of dirt from both the location.



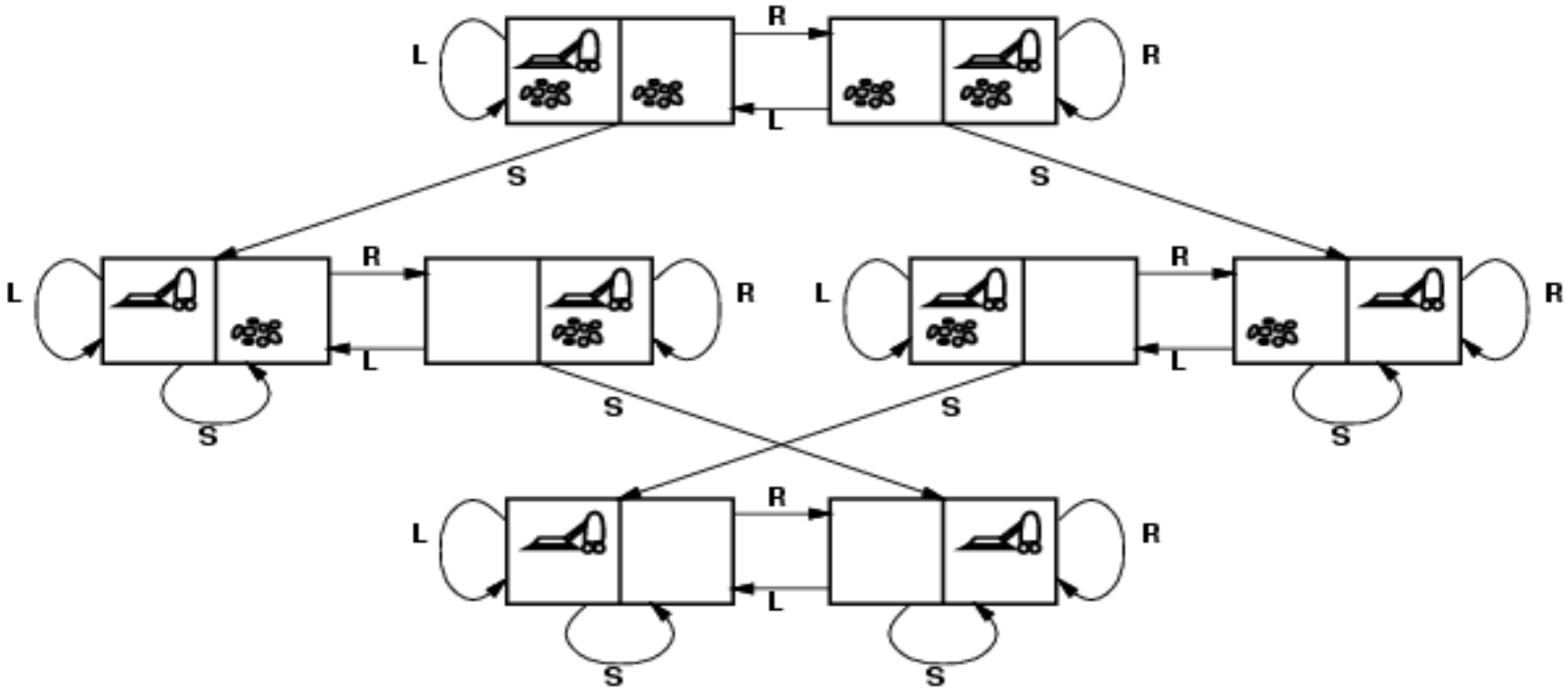
Problem Formulation – Vacuum Cleaner

Problem Formulation :

- The state, operators, goal test and path cost as below :
  1. **States** : One of the eight states (integer dirt and robot location)
  2. **Initial state** : Dirt in both locations and the vacuum cleaner in one of them
  3. **Set of Rules** : L (LEFT), R(RIGHT), S(SUCK)
  4. **Goal Test** : No dirt in any location.
  5. **Path Cost** : Let each action cost be 1 unit.

Problem Formulation – Vacuum Cleaner

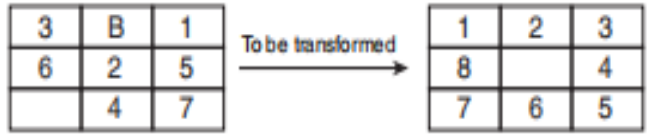
- A state space is drawn such that it represents each state perfectly indicating the presence or absence of dirt in each of the locations.



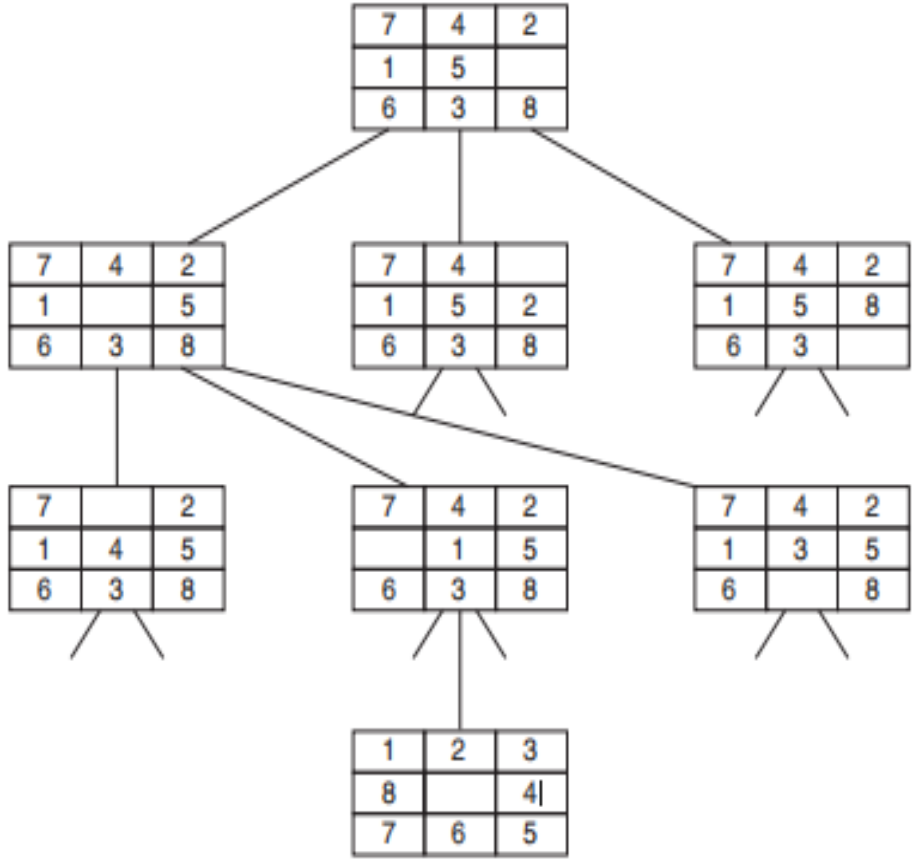
Problem formulation – Eight Tile Puzzle

Problem Definition :

- The eight tile puzzle consist of a 3 × 3 square frame board which holds eight movable tiles numbered 1 to 8.
- One square is empty, allowing the adjacent tiles to be shifted.
- Find a sequence of tile movements that leads from a starting configuration to a goal configuration.



The states of eight-tile puzzle are the different permutations of the tiles within frame.
Let's do a standard formulation of this problem now.



Problem formulation – Eight Tile Puzzle

Problem Formulation :

- **States:** It specifies the location of each of the eight tiles and the blank in one of the nice squares.
- **Initial state:** Any state can be designated as the initial state.
- **Goal:** Many goal configurations are possible.
- **Set of Rules / Legal moves :** They generate legal states that result from trying the four actions:
  - Blank moves left
  - Blank moves right
  - Blank moves up
  - Blank moves down
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

Problem Formulation : Missionaries & Carnivals

Problem Definition :

- Three missionaries and three cannibals are on one side of a river that they wish to cross.

  3M3C –> River –> 3M3C

- A boat is available that can hold at most two people and at least one.
- You must never leave a group of missionaries outnumbered by cannibals on the same bank.
- Find an action sequence that brings everyone safely to the opposite bank.

Missionaries and Cannibals

Problem Formulation :

- States : Configuration of missionaries and cannibals and boat on each side of river.
- Initial State : 3 missionaries, 3 cannibals and the boat are on the near bank
- Rules : Move boat containing some set of occupants across the river (in either direction) to the other side.
- Goal state : Move all the missionaries and cannibals across the river.
- Path Cost : When number of cannibals changes is 6 and number of missionaries change are 4.
- Constraint: Missionaries can never be out numbered by cannibals on either side of river, or else the missionaries are killed.

Solution steps:

SI = [3M, 3C, L]

State of Left Side of river	*Action*	*State of Right side of river*	*Score*
[3M, 1C, L]	→ Send 2C	[0M, 2C, R]	6
[3M, 2C, L]	← Send 1C	[0M, 1C, R]	6
[3M, 0C, L]	→ Send 2C	[0M, 3C, R]	6
[3M, 1C, L]	← Send 1C	[0M, 2C, R]	4
[1M, 1C, L]	→ Send 2M	[2M, 2C, R]	5 average taken = (4 + 6)/2
[2M, 2C, L]	→ Send 2M	[2M, 1C, R]	4
[0M, 2C, L]	→ Send 2M	[3M, 1C, R]	6
[0M, 3C, L]	← Send 1C	[3M, 0C, R]	6
[0M, 1C, L]	→ Send 2C	[3M, 2C, R]	6
[0M, 2C, L]	← Send 1C	[3M, 1C, R]	6
[0M, 0C, L]	→ Send 2C	[3M, 3C, R]	Goal state $\sum = 61$

Total cost = 61 units

Design Issues of Search Program

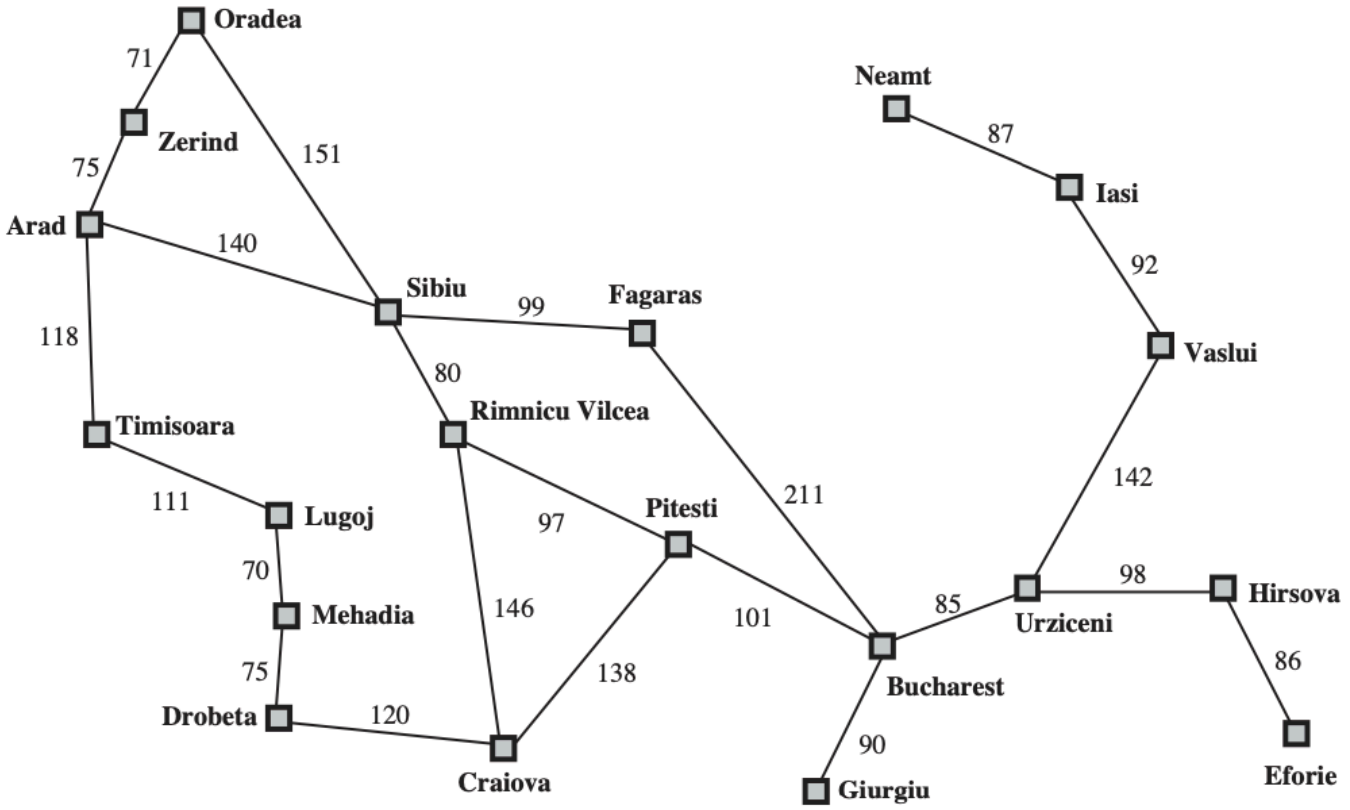
- To choose the most appropriate method for a particular problem. It is necessary to analyse the problem along the several key dimensions.
- Some main features of a problem are listed as follows :
  1. Is the problem decomposable into set of sub problems?
  2. Can the solution step be ignored or undone?
  3. Is the problem universally predictable?
  4. Is a good solution to the problem obvious without comparison to all the possible solutions?
  5. Is the desired solution a state of world or a path to a state?
  6. Is a large number of knowledge absolutely required to solve the problem?
  7. Will the solution of the problem required interaction between the computer and the person?

Searching for Solution

- So far we have talked about how a problem can be looked.
- Having formulated some problems, we now need to solve them.
- A solution is an action sequence, so search algorithms work by considering various possible action sequences.
- The possible action sequences starting at the initial state form a search tree with the initial state at the root.
- The branches are actions and the nodes correspond to states in the state space of the problem.

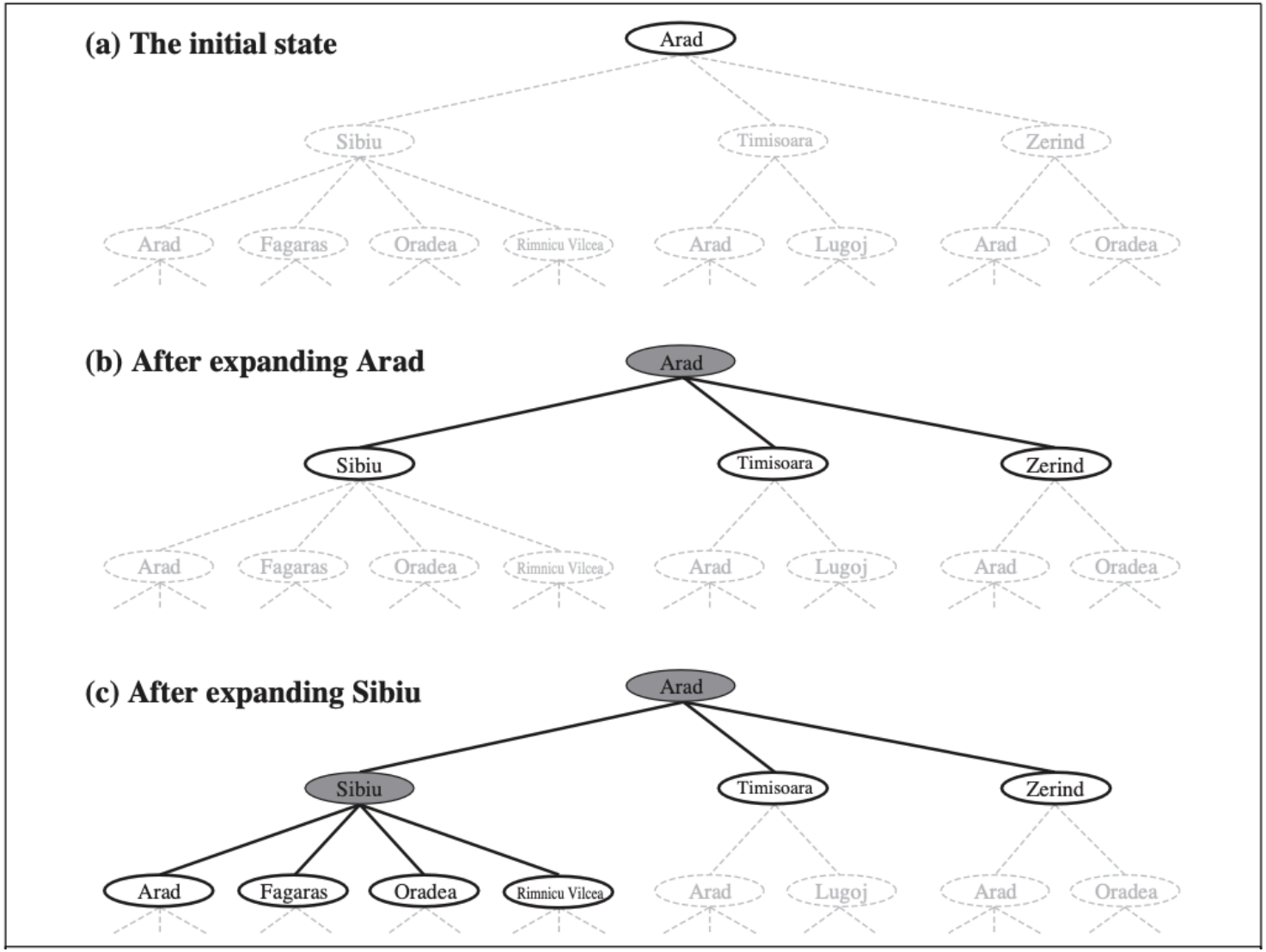
Searching for Solution

- Find a route from Arad to Bucharest.
- The root node of the tree corresponds to the initial state, expand through nodes until either a solution is found or there are no more states to expand.



Searching for Solution

- Repeated state
- Loopy path
- Redundant path



Searching for Solution

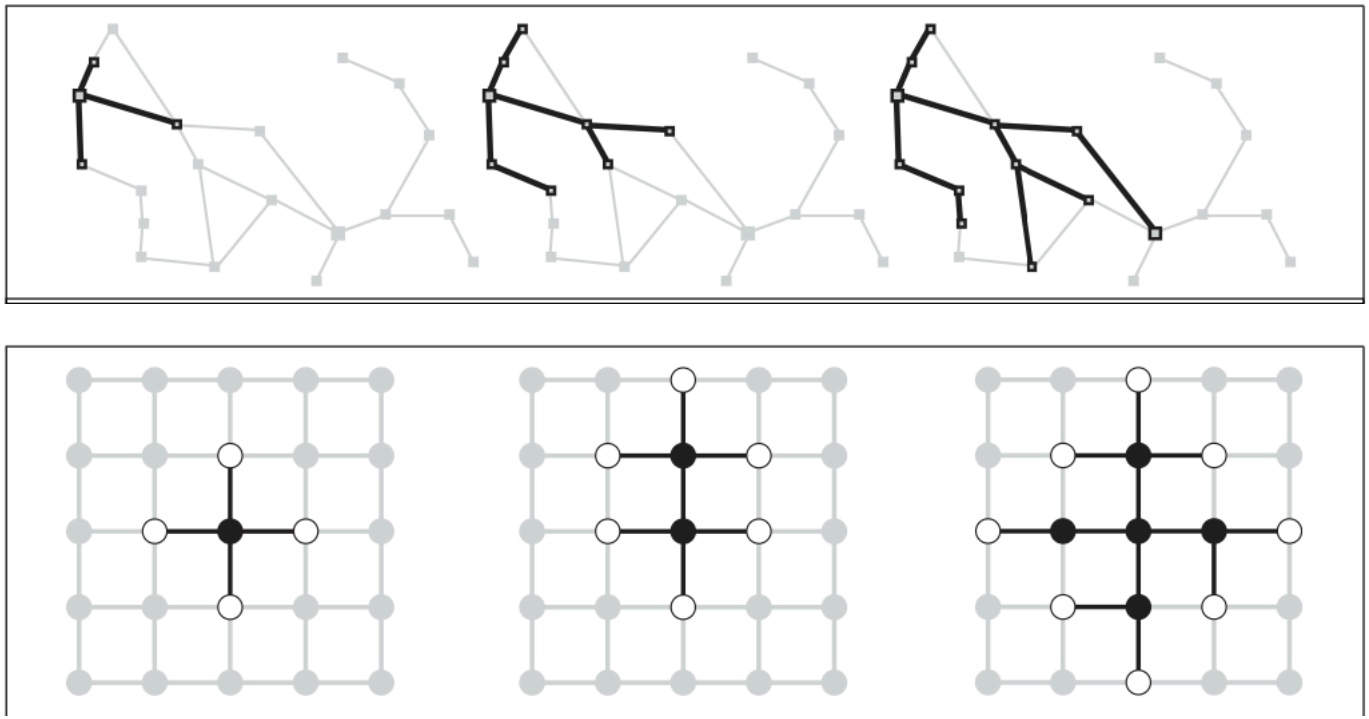
function TREE-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
  initialize the frontier using the initial state of *problem*
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

Searching for Solution

- A sequence of search trees generated by a graph search
- The separation property of GRAPH-SEARCH, The frontier (white nodes) always separates the explored region of the state space (black nodes) from the unexplored region (gray nodes).

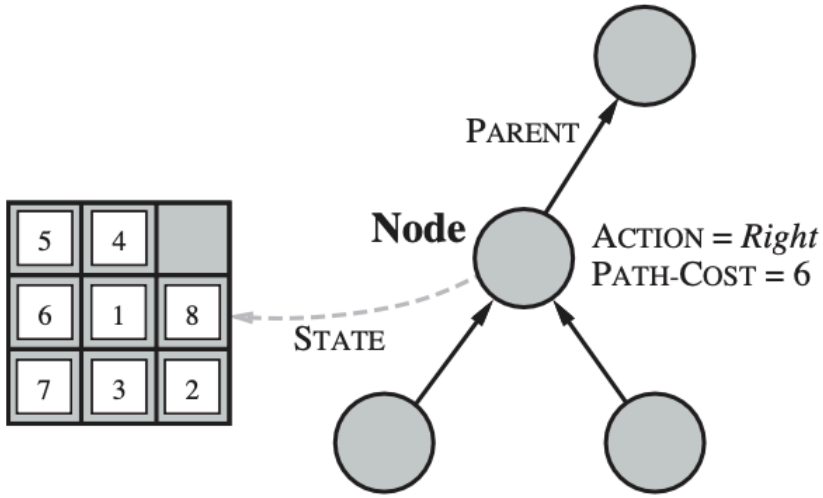


Searching for Solution

- Search algorithms require a data structure to keep track of the search tree that is being constructed.
- For each node n of the tree, we have a structure that contains State, Parent, Action and Path cost.
- There are many ways to represent nodes. Assume that the node is the data structure with the following five components :
  - The state in the state space to which the node corresponds.
  - The node in the search tree that generated this node, this is called as parent node.
  - The operator that was applied to generate the node.
  - The path cost of the path from initial node to the node.
    - Date type : Node
    - Components : State, parent node, action, depth, path-cost.

Searching for Solution

- Queue is used so that search algorithm can easily choose the next node to expand according to its preferred strategy.



function CHILD-NODE(*problem*, *parent*, *action*) **returns** a node
  return a node with
    STATE = *problem*.RESULT(*parent*.STATE, *action*),
    PARENT = *parent*, ACTION = *action*,
    PATH-COST = *parent*.PATH-COST + *problem*.STEP-COST(*parent*.STATE, *action*)

Searching for Solution

- Measuring problem-solving performance
  - **Completeness:** A search algorithm is complete if it finds a solution whenever one exists.
  - **Optimality:** A search algorithm is optimal if it returns a minimum-cost path whenever a solution exists
  - **Time complexity:** How long does it take to find a solution.
  - **Space complexity:** How much memory is needed to perform the search.
- Complexity is measured by ***b*** branching factor (maximum number of successors of any node), ***d*** depth of the shallowest goal node (the number of steps along the path from the root), ***m*** maximum length of any path in the state space.
- Time is often measured in terms of the number of nodes generated during the search, and space in terms of the maximum number of nodes stored in memory.

Search Strategies

- Searching is the process to find the solution for a given set of problems. This in AI can be done by using either uninformed searching strategies or informed searching strategies.
- Uninformed Search / Blind Search :
  - This method has no information about the number of steps or path cost from the current state to goal state.
  - Not aware that search process is moving in the direction of the goal or in the opposite direction.
  - It only checks for the state generated is goal or not. If so it will stop otherwise continues to generate next state.
- Informed Search :
  - This method knows whether one goal state is more promising than other is called a heuristic / informed search techniques.
  - They are aware if the search is moving in the right direction or opposite direction
  - It can choose the non goal state that is most closest to the goal as the next state.
  - If the state generated is a goal state, the search process stops else continues to generate next state in a state space.