Artificial Intelligence

Course Code: 23CS5PCAIP

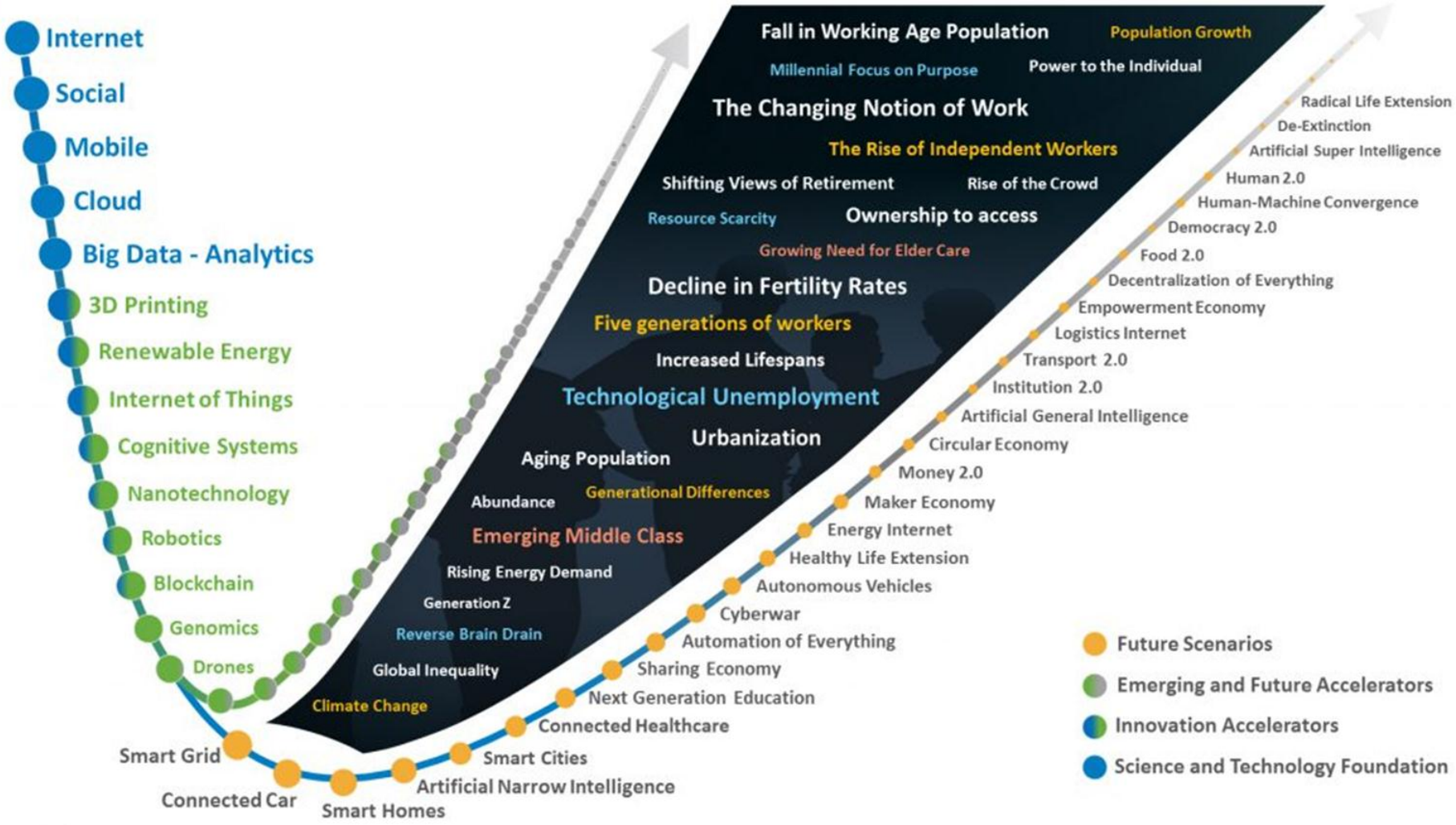
Syllabus

Unit	Content	Books – Chapter number
Unit 1	Introduction Intelligent Agents Solving Problems by Searching	Book 1: chapter 1:1.1-1.3 Chapter 2: 2.1-2.4 Chapter 3: 3.1-3.3
Unit 2	Uninformed Search Strategies Informed Search and Exploration	Chapter 3: 3.4-3.6 Chapter 4: 4.1-4.2
Unit 3	Logical Agents First Order Logic	Chapter 7: 7.1-7.7 Chapter 8:8.1-8.3
Unit 4	Inference in First Order Logic	Chapter 9: 9.1-9.5
Unit 5	Uncertainty Probabilistic Reasoning	Chapter 13: 13.1-13.6 Chapter 14: 14.1-14.5

Why AI?

https://www.youtube.com/watch?v=FscTCQdwLyc

https://www.youtube.com/watch?v=oV74Najm6Nc


Internet
Social
Mobile
Cloud
Big Data - Analytics
3D Printing
Renewable Energy
Internet of Things
Cognitive Systems
Nanotechnology
Robotics
Blockchain
Genomics
Drones
Smart Grid
Connected Car
Smart Homes
Artificial Narrow Intelligence
Smart Cities
Connected Healthcare
Next Generation Education
Sharing Economy
Automation of Everything
Cyberwar
Autonomous Vehicles
Healthy Life Extension
Energy Internet
Maker Economy
Money 2.0
Circular Economy
Artificial General Intelligence
Institution 2.0
Transport 2.0
Logistics Internet
Empowerment Economy
Decentralization of Everything
Food 2.0
Democracy 2.0
Human-Machine Convergence
Human 2.0
Artificial Super Intelligence
De-Extinction
Radical Life Extension
Fall in Working Age Population
Population Growth
Millennial Focus on Purpose
Power to the Individual
The Changing Notion of Work
The Rise of Independent Workers
Shifting Views of Retirement
Rise of the Crowd
Resource Scarcity
Ownership to access
Growing Need for Elder Care
Decline in Fertility Rates
Five generations of workers
Increased Lifespans
Technological Unemployment
Urbanization
Aging Population
Abundance
Generational Differences
Emerging Middle Class
Rising Energy Demand
Generation Z
Reverse Brain Drain
Global Inequality
Climate Change
Future Scenarios
Emerging and Future Accelerators
Innovation Accelerators
Science and Technology Foundation

Definition

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

- A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.
- The computer would need to possess the following capabilities:
- NATURAL LANGUAGE • **natural language processing to enable it to communicate successfully in English;**
- PROCESSING KNOWLEDGE • **knowledge representation to store what it knows or hears;**
- REPRESENTATION AUTOMATED • **automated reasoning to use the stored information to answer questions and to draw new conclusions;**
- MACHINE LEARNING • **machine learning to adapt to new circumstances and to detect and extrapolate patterns.**
- To pass the total Turing Test, the computer will need
  - COMPUTER VISION • **computer vision to perceive objects, and**
  - ROBOTICS • **robotics to manipulate objects and move about.**

Thinking humanly: The cognitive modeling approach

- Incorporates neuro-physiological evidence into computational models.
- For example, Allen Newell and Herbert Simon, who developed GPS, the “General Problem Solver” (Newell and Simon, 1961), were not content merely to have their program solve problems correctly. They were more concerned with comparing the trace of its reasoning steps to traces of human subjects solving the same problems.

Thinking rationally: The “laws of thought” approach

- The Greek philosopher Aristotle was one of the first to attempt to codify “right thinking,” that is, irrefutable reasoning processes. His **syllogisms provided patterns for argument structures** that always yielded correct conclusions when given correct premises
- For example, “Socrates is a man; all men are mortal; therefore,
- Socrates is mortal.”
- These laws of thought were supposed to govern the operation of the mind; their study initiated the field called **logic.**

Acting rationally: The rational agent approach

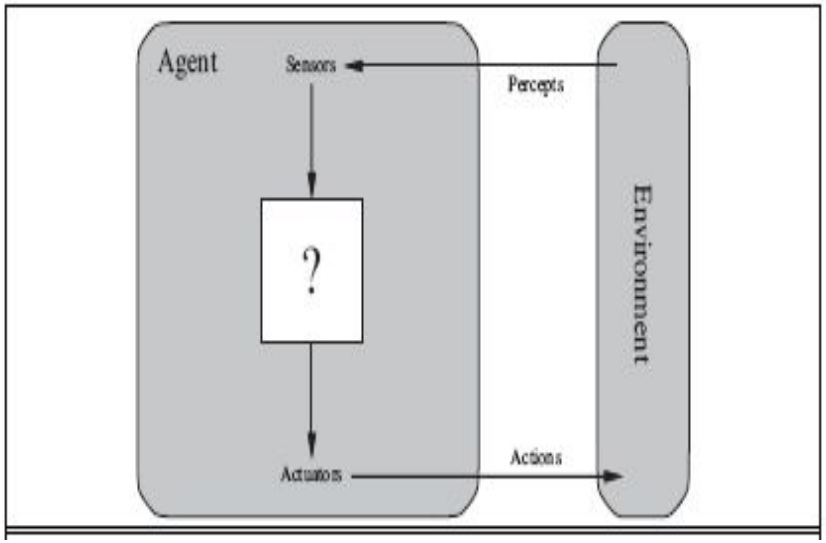
- An **agent is just something that acts (*agent comes from the Latin agere, to do). Of course,*** all computer programs do something, but computer agents are expected to do more:
  - operate autonomously,
  - perceive their environment, persist over a prolonged time period,
  - adapt to change, and
  - 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.

Chapter 2

INTELLIGENT AGENTS 2.1-2.4

2.1 Agents and Environments

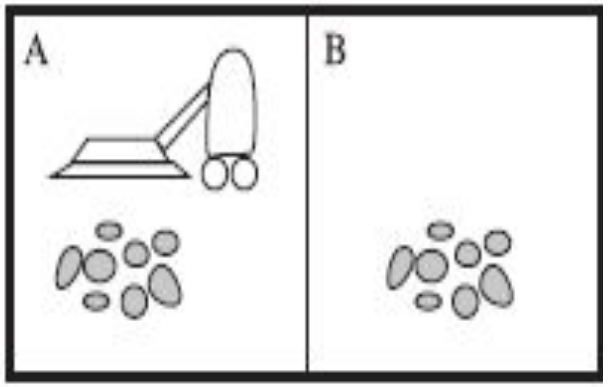
An **agent is anything that can be viewed as perceiving its environment through sensors and** acting upon that environment through **actuators.**



Definitions

- The term **percept to refer to the agent's perceptual inputs at any given instant.**
- **An** agent's **percept sequence is the complete history of everything the agent has ever perceived.**
- An agent's behavior is described by the **agent function that maps any given percept sequence to an action.**
- **Agent Program**

Example- Vacuum Cleaner



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
⋮	⋮

Agent Function for Vacuum Cleaner Problem

```
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
```

Other Topics

- Good behaviour – The Concept of Rationality
- The Nature of Environments
- The Structure of an Agents

2.2 The concept of rationality

- A **rational agent is one that does the right thing**
- When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives.
- This sequence of actions causes the environment to go through a sequence of states.

- Desirability is captured by a **performance measure that** evaluates any given sequence of environment states.
- *it is better to design performance measures according to what **one actually wants** in the environment, rather than according to how **one thinks the agent should behave**.*

Rationality 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.

- *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.*
- *Is Vacuum cleaner agent rational?*

IS OMNISCIENT AND RATIONALITY SAME?

- An Rational agent knows the *actual outcome of its actions and can act accordingly; but omniscience is* impossible in reality.

  Example:

- I am walking along the Champs Elys´ees one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner, and before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read "Idiot attempts to cross street."

Important aspects

- Doing actions *in order to modify future percepts—sometimes called* ***information gathering—*** is an important part of rationality
- A second example of information gathering is provided by the **exploration that must be undertaken by** a vacuum-cleaning agent in an initially unknown environment.

Why information gathering is important?

A rational agent not only requires to **gather information** but also to **learn** as much as possible from what it perceives.

Dung Beetle

Sphex Wasp

Importance of autonomy

- If an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks **autonomy.**
- **A rational agent should be** autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.

2.3 The nature of environments

- Correct problem identification is **task environments, which are essentially** the "problems" to which rational agents are the "solutions."
- In designing an agent, the first step must always be to specify the task environment as fully as possible.
- ***PEAS (Performance, Environment, Actuators, Sensors)***

Example

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

New intelligent agent- softbots

- **Software agents (or software robots or softbots) exist in** rich, unlimited domains.
- Imagine a softbot Web site operator designed to scan Internet news sources and show the interesting items to its users, while selling advertising space to generate revenue.

Properties of task environment -1

- **Fully observable vs partially observable - If an agent's sensors give it access to the** complete state of the environment at each point in time, then we say that the task environment is fully observable.
  - *Unobservable*

Properties of task environment -2

- **Single agent vs multiagent**
  - Competitive, cooperative
- **Deterministic vs stochastic**
  - **If the next state of the environment is completely** determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic.
  - Uncertain, nondeterministic

Properties of task environment -3

- **Episodic vs sequential - In an episodic task environment, the agent's experience is** divided into atomic episodes. In each episode the agent receives a percept and then performs a single action.
- **Static vs dynamic - If the environment can change while an agent is deliberating, then** we say the environment is dynamic for that agent; otherwise, it is static.
  - Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing.
  - Semi dynamic

Properties of task environment -4

- **Discrete vs continuous - The discrete/continuous distinction applies to the *state of the*** environment, to the way *time is handled, and to the percepts and actions of the agent.*
- **Known vs unknown - This distinction refers not to the environment i**tself but to the agent's (or designer's) state of knowledge about the "laws of physics" of the environment.
  - *Environment generator*

Examples

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

2.4 The structure of agents

- *agent = architecture + program .*
- The agent programs take the current percept as input from the sensors and return an action to the actuators.

Table driven agent

```
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
```

Is Table driven agent a good approach?

Even the lookup table for chess—a tiny, well-behaved fragment of the real world—would have at least 10^{150} entries.

Four basic kinds of agent programs

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents

Type of AI Agent	Quick Explanation
Simple Reflex	These agents react immediately to their surroundings based on pre-defined rules
Model-based Reflex	They are similar to simple reflex agents but maintain an internal model of their environment. This model allows them to handle situations where not everything is directly observable
Goal-based	These agents have specific goals in mind and plan their actions accordingly
Utility-based	These AI agents go beyond goals and consider the desirability of different outcomes
Learning	These agents can enhance their performance over time by learning from experience

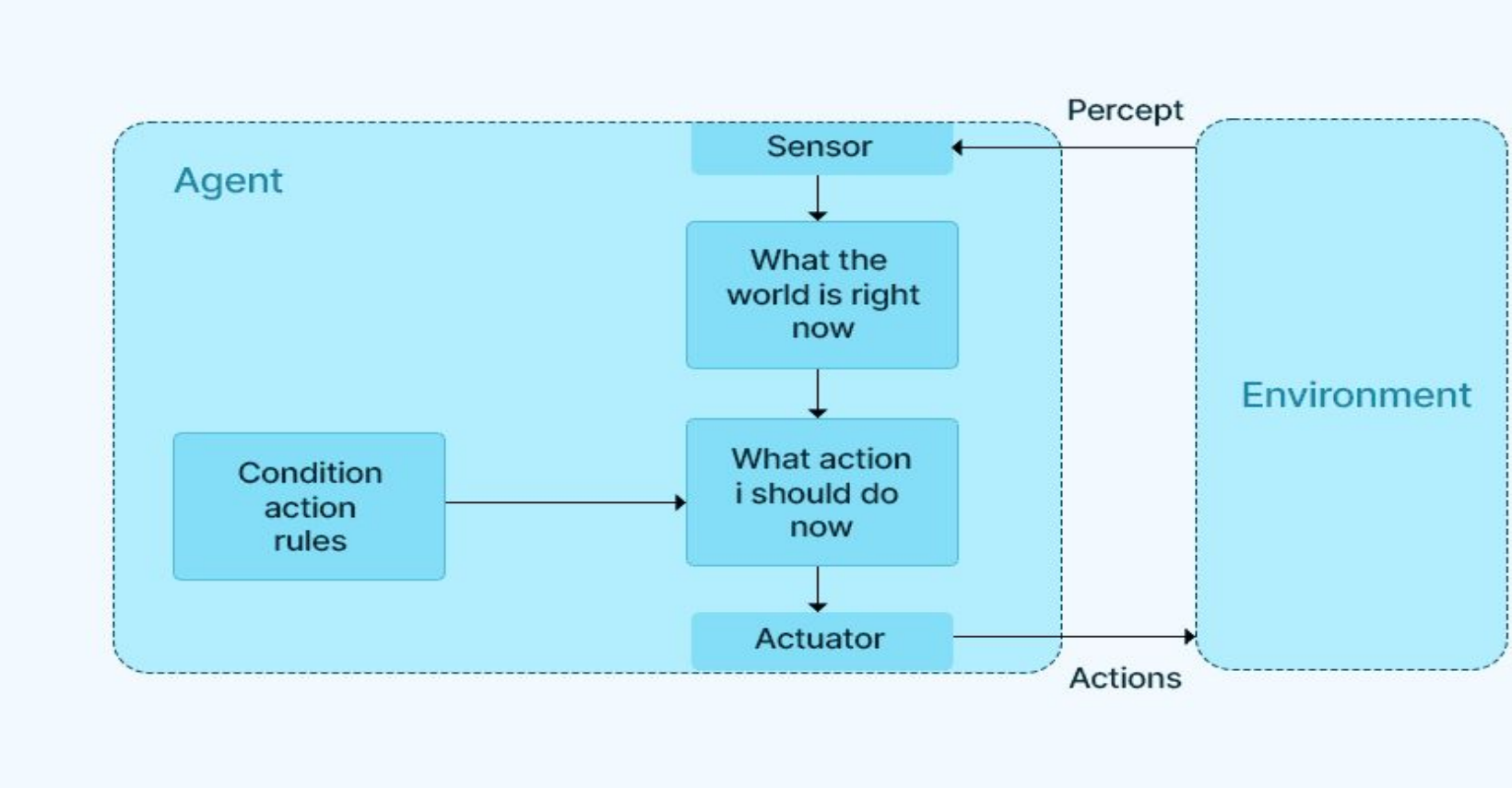
SIMPLE REFLEX

- The simplest kind of agent is the **simple reflex agent. These agents select actions on the basis** of the *current percept, ignoring the rest of the percept history.*
- "The car in front is braking." Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a **condition–action rule, written as**
  - **if *car-in-front-is-braking then initiate-braking.***
- Humans also have many such connections, some of which are learned responses (as for driving) and some of which are innate reflexes (such as blinking when something approaches the eye).

Reflex Vacuum Agent

```
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
```

Schematic Diagram



Function

```
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
```

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 Agent

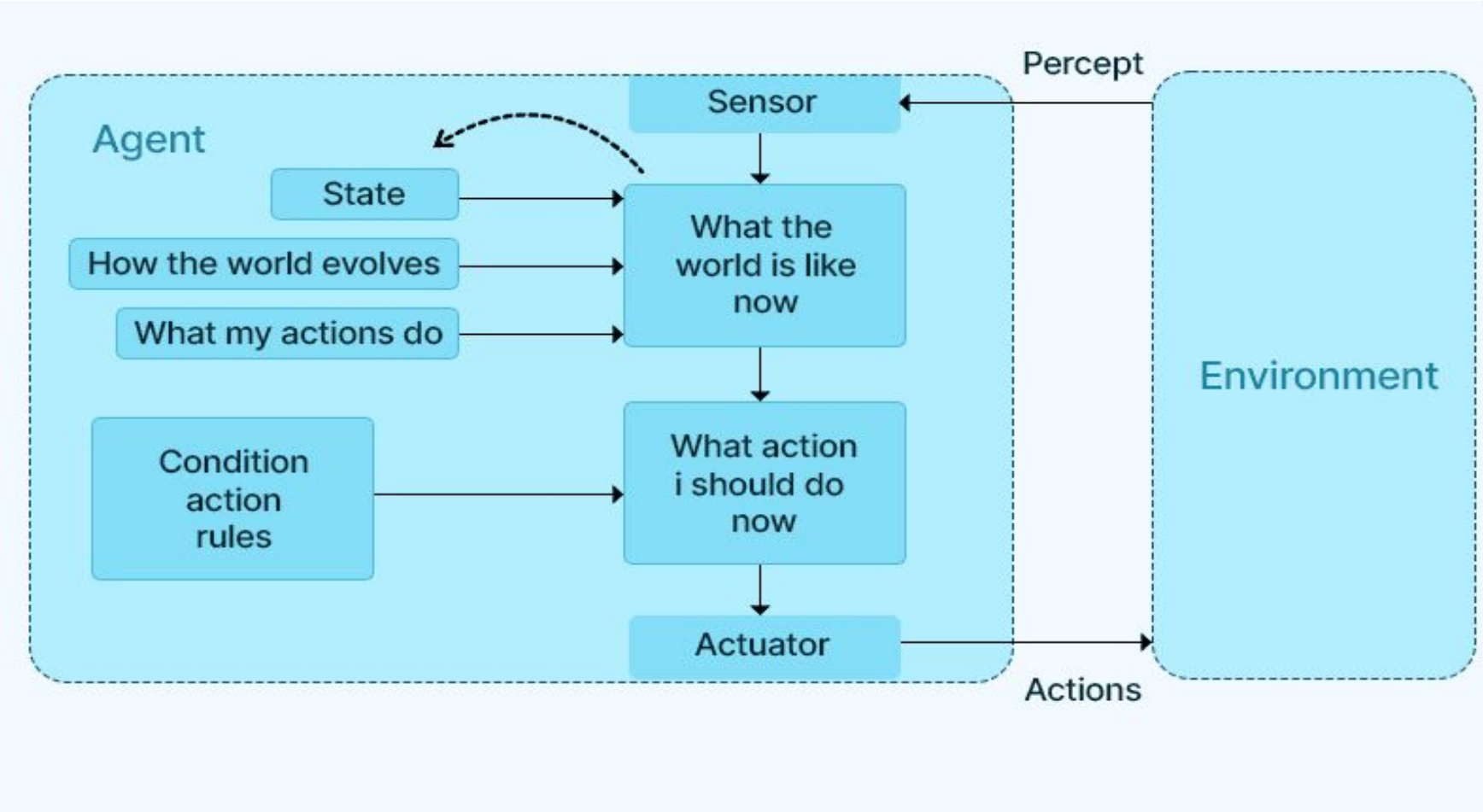
- The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now.*
- *The agent should maintain some sort of* ***internal*** **state that depends on the percept history and thereby reflects at least some of the unobserved** aspects of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

AI agents rely on pre-programmed rules and don't exhibit true learning capabilities.

Agents functionality

- First, we need some information about **how the world evolves independently of the agent**
- Second, we need some information about **how the agent's own actions affect the world**

Schematic Diagram

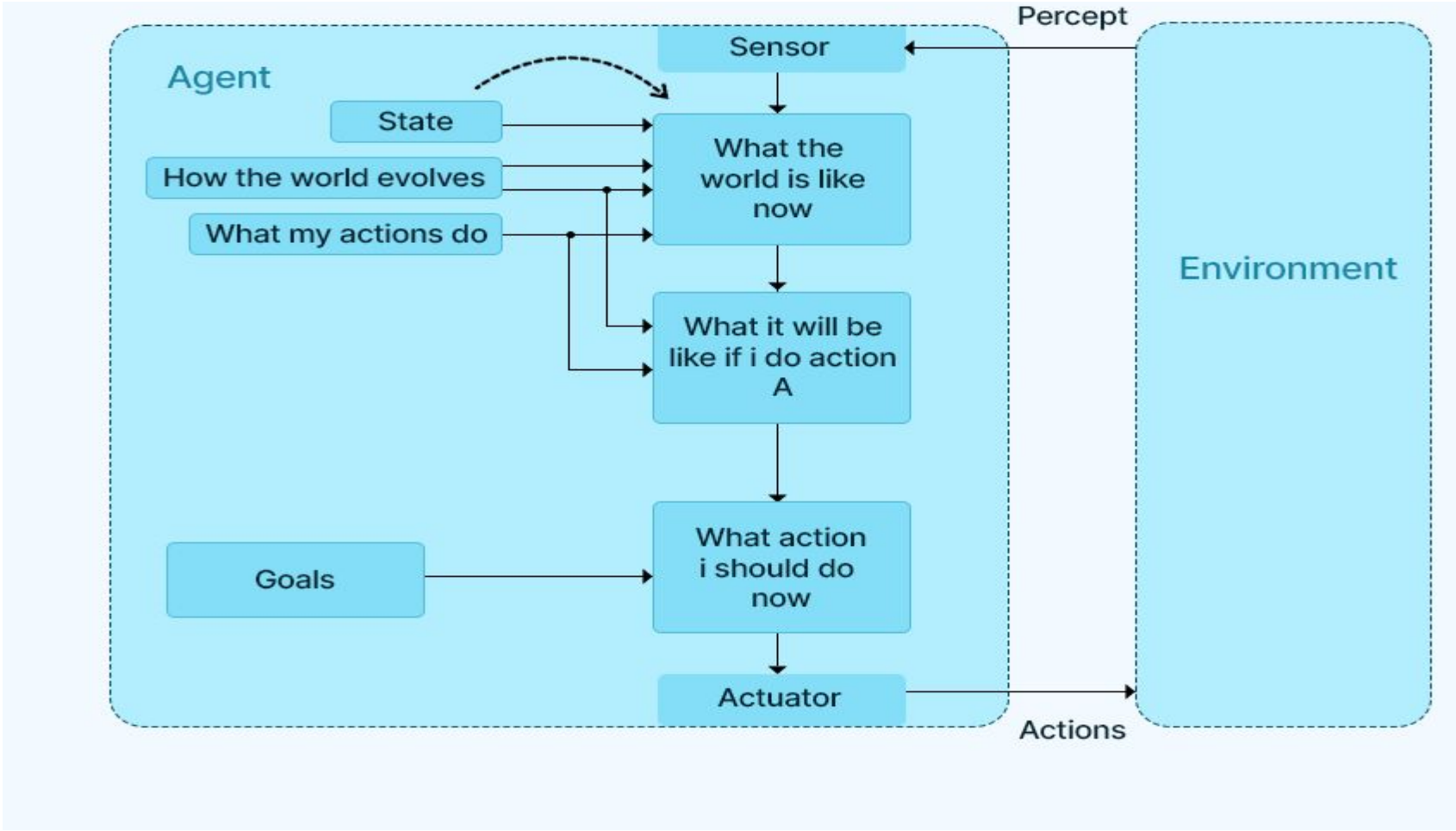


Function

```
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



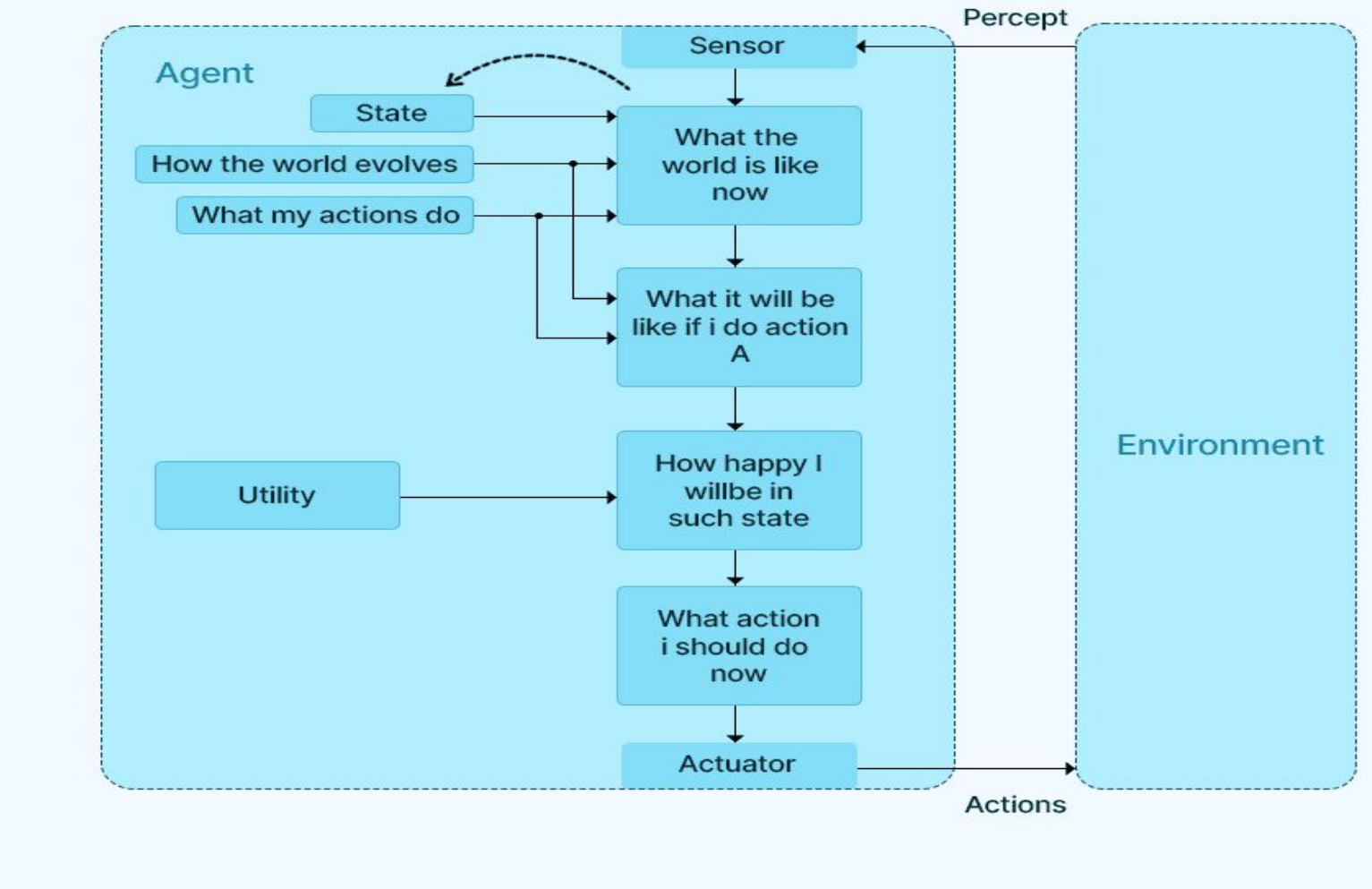
Working

- Agent needs current state description, along with some sort of **goal information that describes** situations that are desirable – to take decision
  - Search and planning are the sub fields
- Although 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.

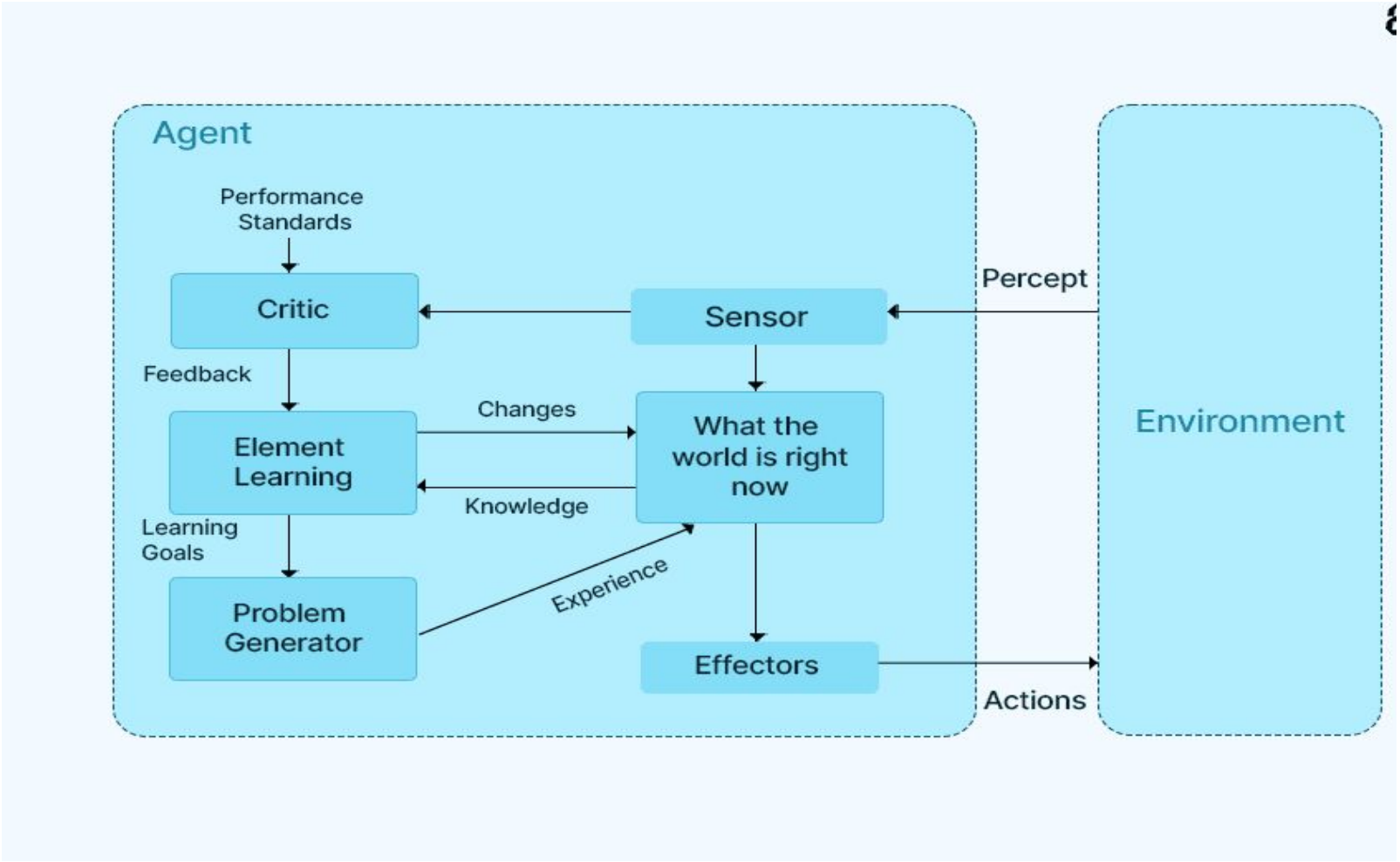
Utility based agent

- An agent's **utility function is essentially an internalization** 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.
- Decision-making agents that must handle the uncertainty inherent in stochastic or partially observable environments.

Schematic Diagram



Learning Agent



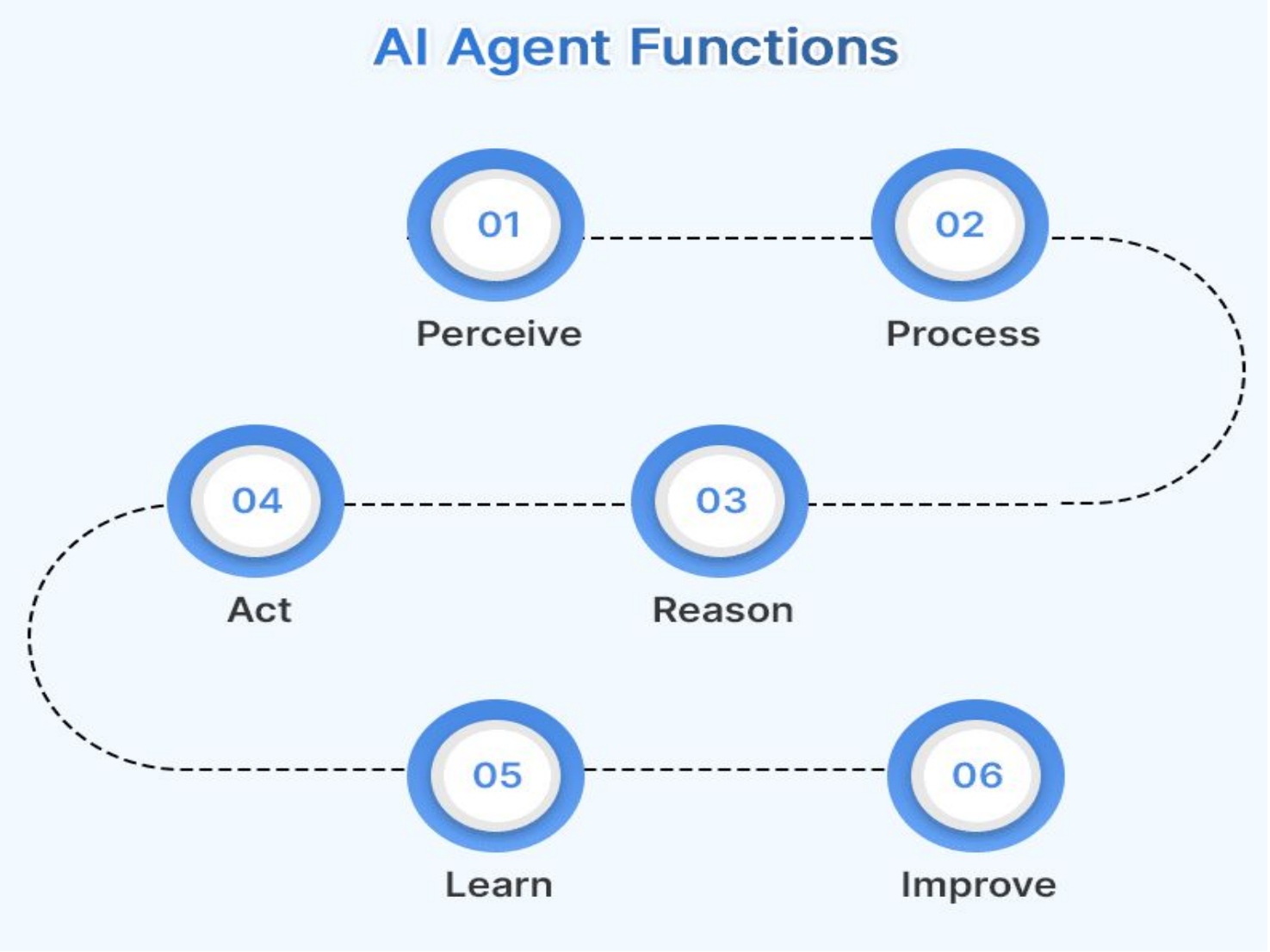
Learning Agent

- The most important distinction is between the **learning element,** which is responsible for making improvements, and the **performance element,** which is responsible for selecting external 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

- **Problem generator is responsible** for suggesting actions that will lead to new and informative experiences.

Functions of an AI Agent



End of chapter 2

End of Class

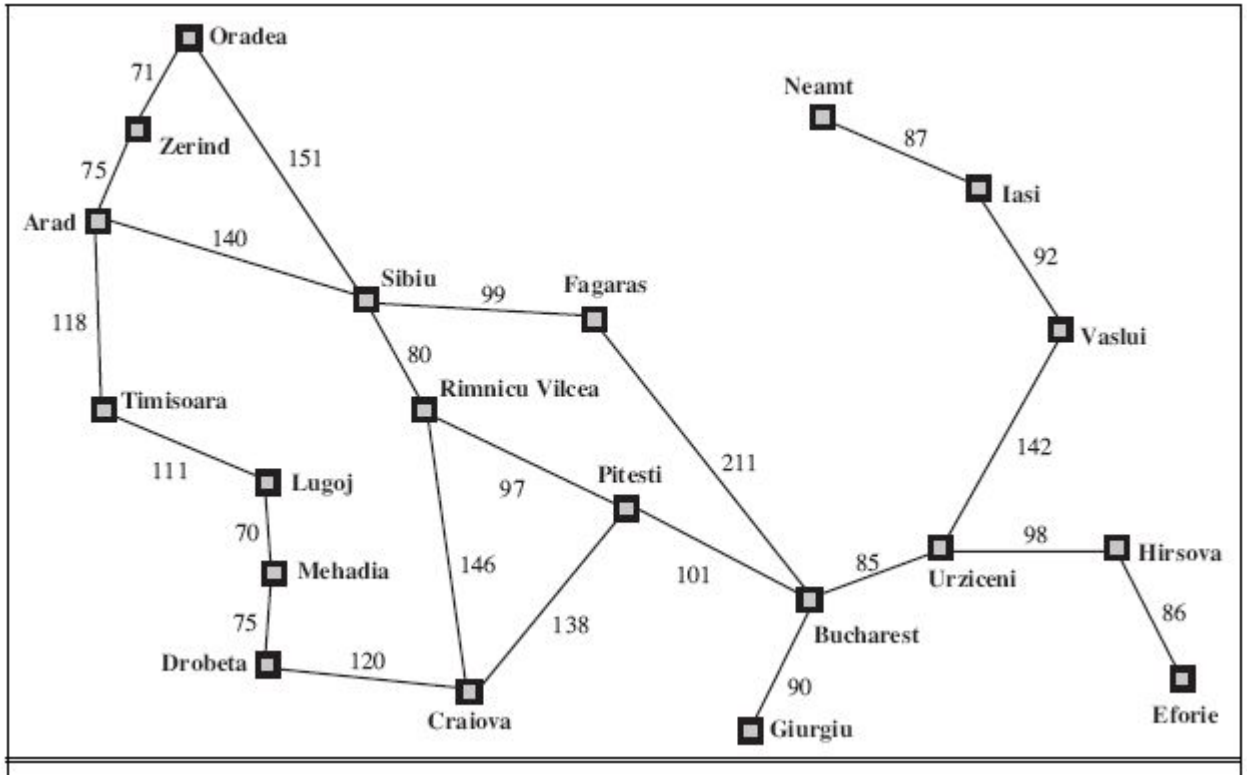
Chapter 3

SOLVING PROBLEMS BY SEARCHING 3.1-3.3

3.1 Problem solving agents

- In problem solving, **Goal formulation,** based on the **current situation and the agent's performance measure** is the first step.
- **Problem formulation** is the process of deciding what **actions** and **states** to consider, given a goal.
- Imagine an agent in the city of Arad, Romania, enjoying a touring holiday. The agent's performance measure contains many factors: it wants to improve its suntan, improve its Romanian, take in the sights, enjoy the nightlife (such as it is), avoid hangovers, and so on. The decision problem is a complex one involving many tradeoffs and careful reading of guidebooks.

Simplified map of part of Romania



Terms

- If the agent has no additional information—i.e., if the environment is **unknown in the sense —then it is has no choice but to** try one of the actions at random. This is sad situation.
- *An agent with several immediate options of unknown value can decide what to do by first **examining future actions** that eventually lead to states of known value.*

Terms

- The environment is **observable,** so the agent always knows the current state.
- The environment is **discrete,** so at any given state there are only finitely actions to choose from.
- The environment is **deterministic, so each action has exactly one** outcome. Under ideal conditions, this is true for the agent in Romania—it means that if it chooses to drive from Arad to Sibiu, it does end up in Sibiu.

Terms

- The process of looking for a sequence of actions that reaches the goal is called **search.**
- 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.**

Terms

- The agent that is executing the solution sequence it *ignores its percepts* when choosing an action because it knows in advance what they will be.
- An agent that carries out its plans with its eyes closed, so to speak, must be quite certain of what is going on. Control theorists call this an **open-loop system, because ignoring the percepts breaks the** loop between agent and environment.

Simple 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
```

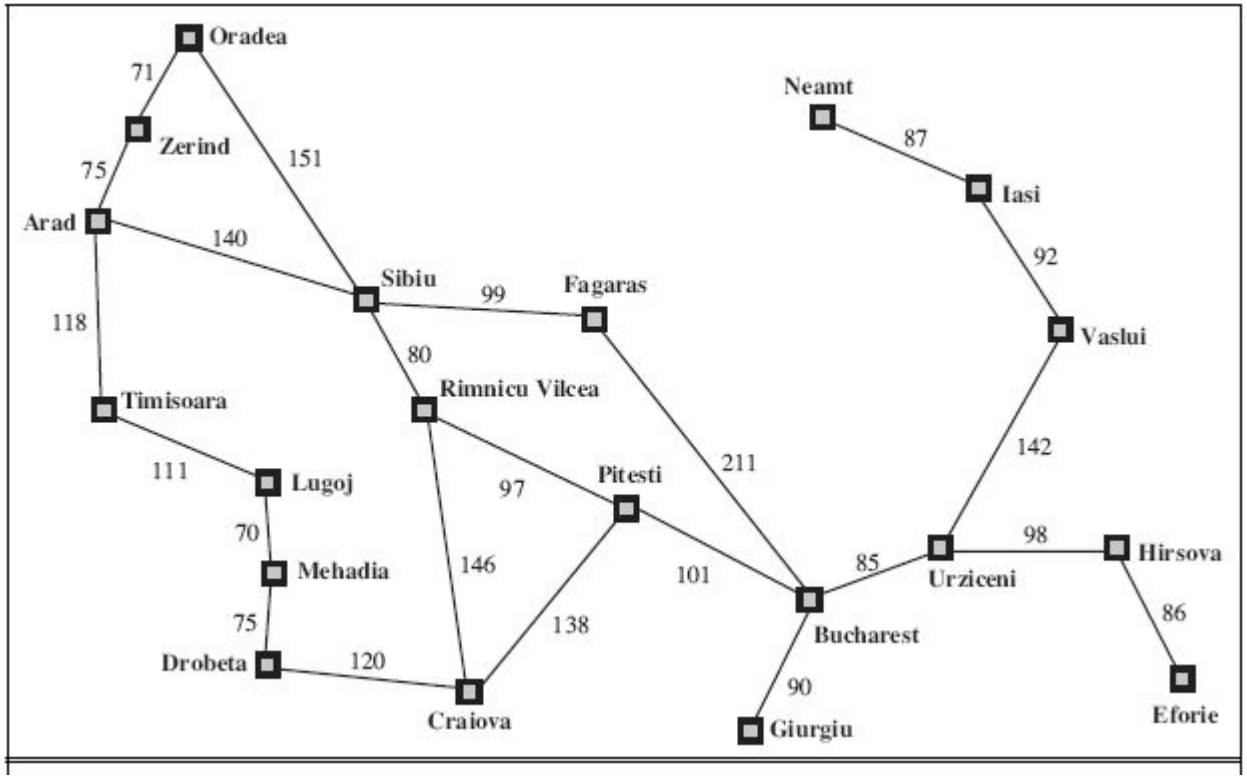
Problem – 5 components

- **INITIAL STATE** • The initial state that the agent starts in. For example, the initial state for our agent in Romania might be described as In(Arad).
- 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(Arad), the applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.
- 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*
  - RESULT(In(Arad),Go(Zerind)) = In(Zerind) .
  - the initial state, actions, and transition model implicitly define the **state space** of the problem
  - Graph n path

Problem – 5 components

- 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.
  - Step cost - The **step cost of taking action** a in state s to reach state s' is denoted by c(s, a, s').

Simplified map of part of Romania



Formulating Problems

- 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.

Abstraction

- Compare the simple state description we have chosen, *In(Arad), to an actual cross country* trip, where the state of the world includes so many things: the traveling companions, the current radio program, the scenery out of the window, the proximity of law enforcement officers, the distance to the next rest stop, the condition of the road, the weather, and so on.
- All these considerations are left out of our state descriptions because they are irrelevant to the problem of finding a route to Bucharest.
- The process of removing detail from a representation is called **abstraction.**

Abstract the action too!

- In addition to abstracting the state description, we must abstract the actions themselves.
- Consumes fuel, generates pollution, and changes the agent (as they say, travel is broadening).

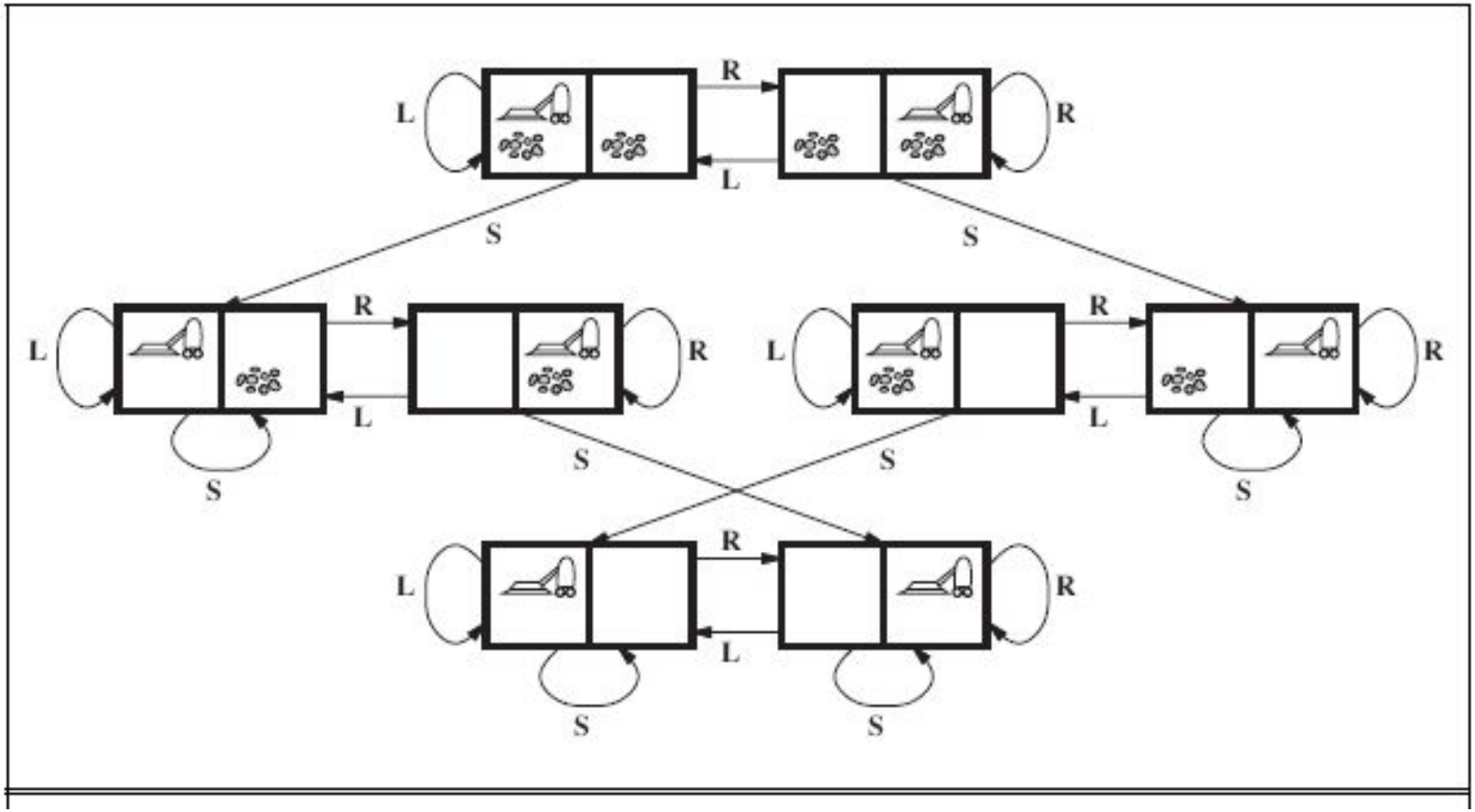
Abstract the action too!

- The choice of a good abstraction thus involves removing as much detail as possible while retaining validity and ensuring that the abstract actions are easy to carry out.

3.2 Example Problems

- Toy problem
- Real world problem

State Space - Vacuum world

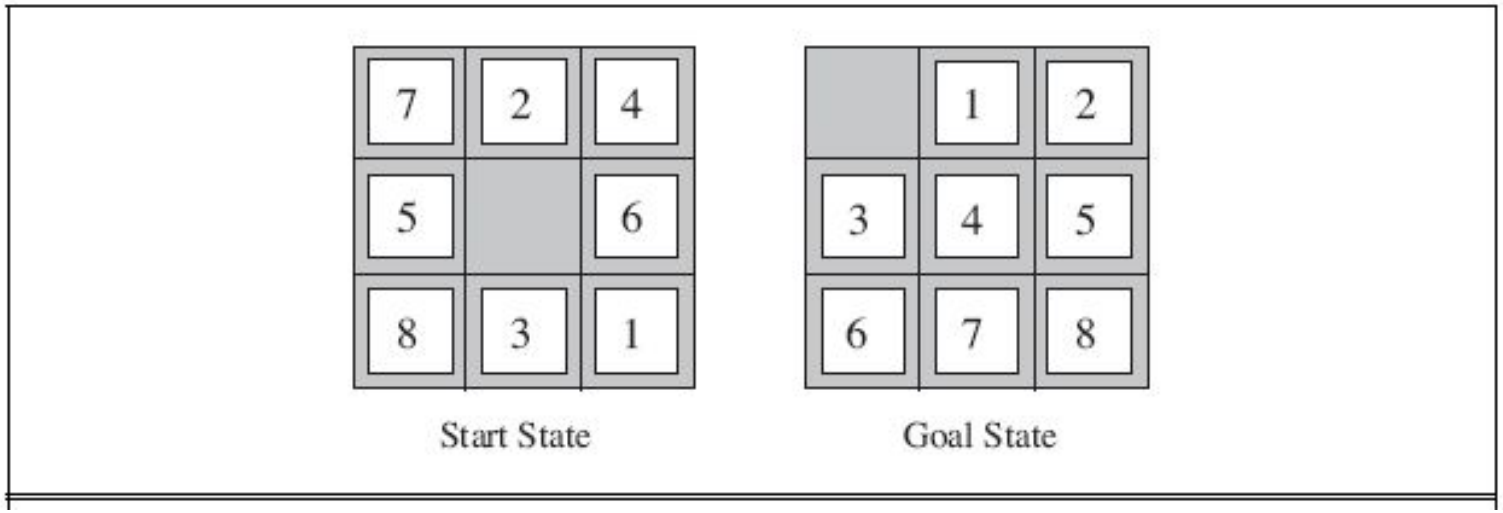


Standard Formulation

- **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 Suck. 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.

End of class

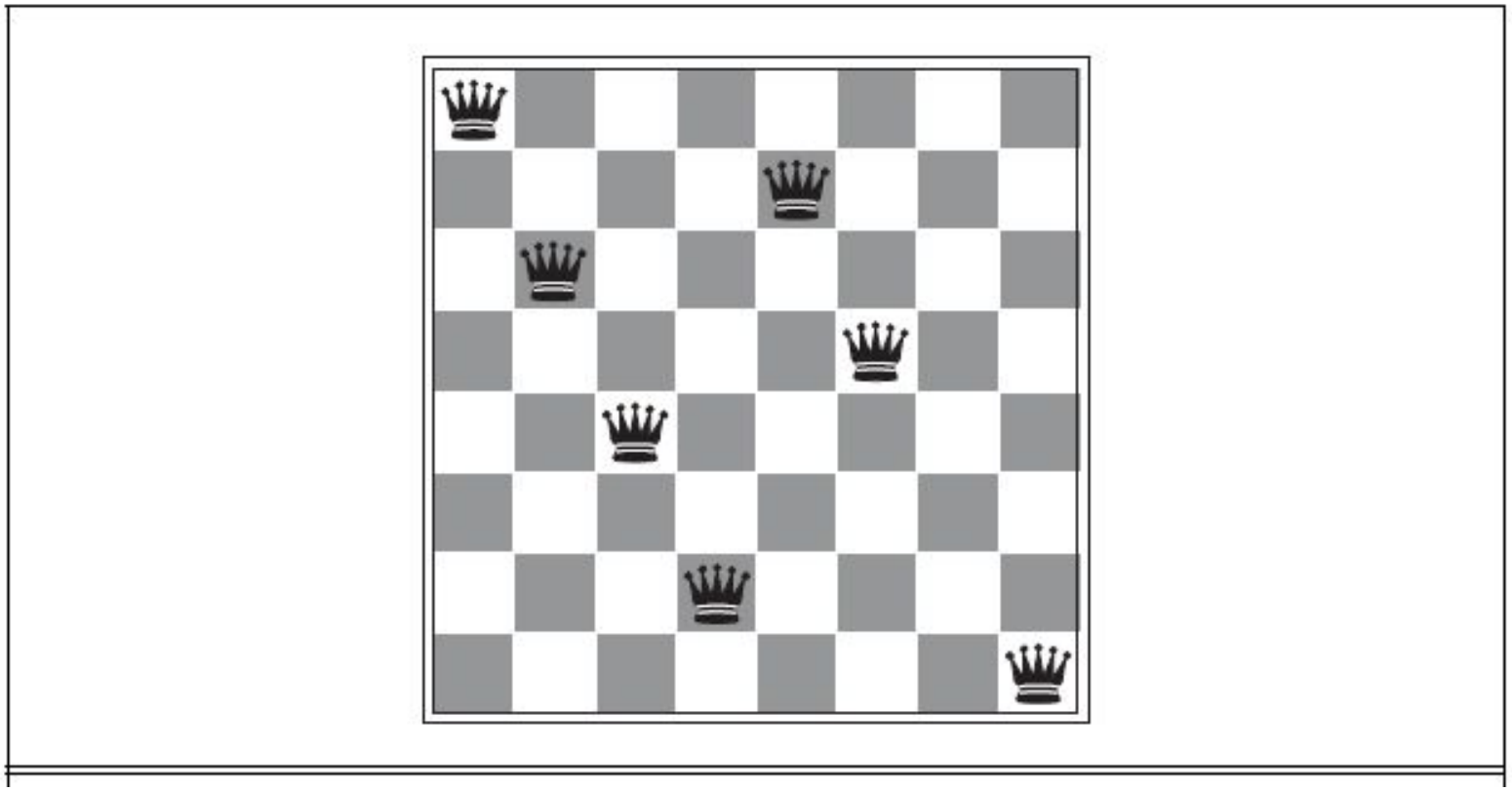
The **8-puzzle**



Standard Formulation

- **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. Note that any given goal can be reached from exactly half of the possible initial states
- **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; for example, if we apply *Left to the start state the resulting state has the 5 and the blank* switched.
- **Goal test:** This checks whether the state matches the goal configuration (Other goal configurations are possible.)
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

8-queens problem



Standard Formulation

- **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.
- ***States:*** *All possible arrangements of n queens ($0 \leq n \leq 8$), one per column in the leftmost n columns, with no queen attacking another.*
- ***Actions:*** *Add a queen to any square in the leftmost empty column such that it is not attacked by any other queen.*

8-queens problem

- This formulation reduces the 8-queens state space from 3×10^{14} to just 2,057, and solutions are easy to find. On the other hand, for 100 queens the reduction is from roughly 10^{400} states to about 10^{52} states

Real-world problems - route-finding problem

- **States:** Each state obviously includes a location (e.g., **an airport**) and the current time. Furthermore, because the cost of an action (a flight segment) may depend on previous segments, their fare bases, and their status as domestic or international, the state must record extra information about these "historical" aspects.
- **Initial state:** This is specified by the user's query.
- **Actions:** Take any flight from the current location, in any seat class, leaving after the current time, leaving enough time for within-airport transfer if needed.
- **Transition model:** The state resulting from taking a flight will have the flight's destination as the current location and the flight's arrival time as the current time.
- **Goal test:** Are we at the final destination specified by the user?
- **Path cost:** This depends on monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer mileage awards, and so on.

Other Examples

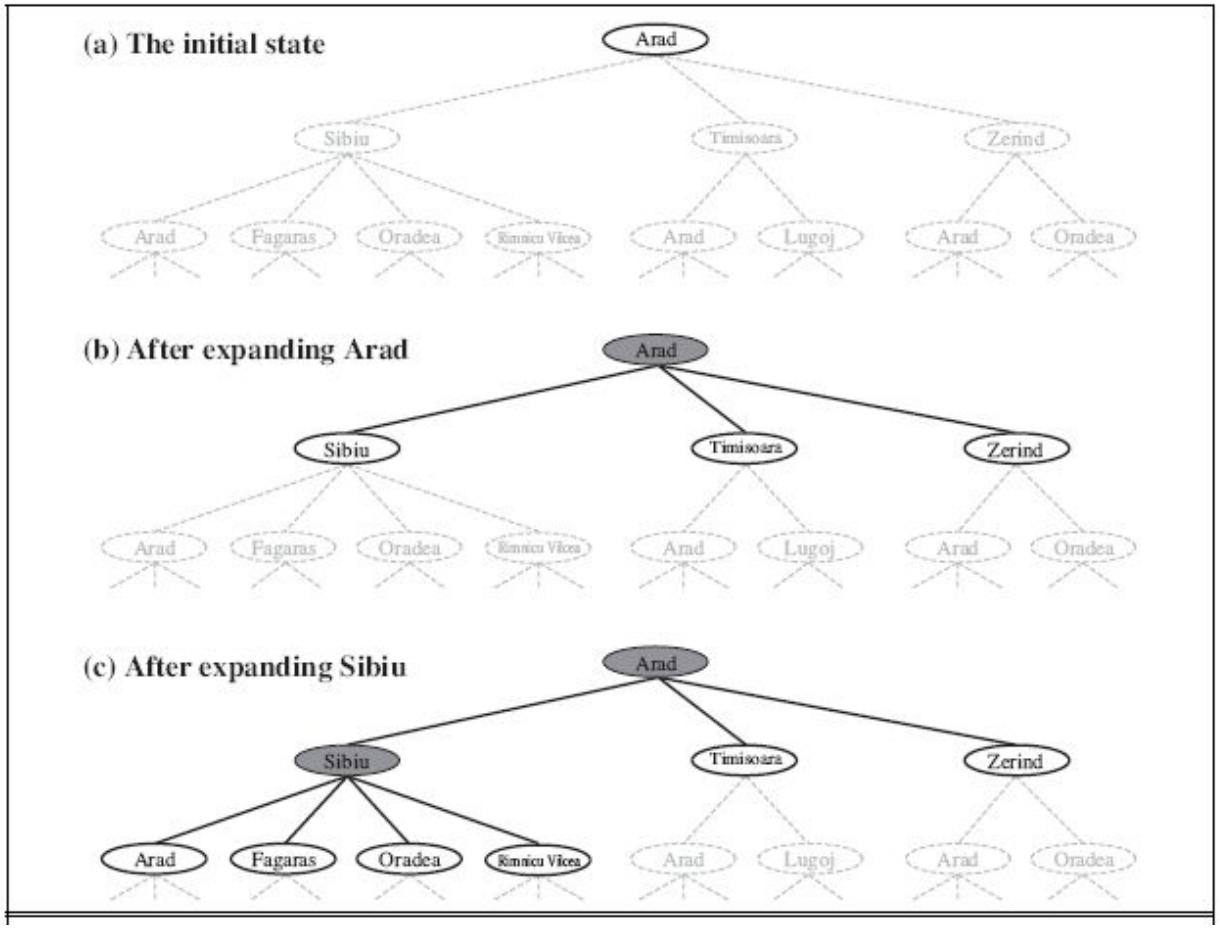
- **Touring problems**
- **Traveling salesperson problem (TSP)**
- **VLSI Layout**
- **Robot navigation**
- **Automatic assembly sequencing**
- **Protein design – right sequence of amino acid, with 3-D protein**

3.3 Searching for Solutions

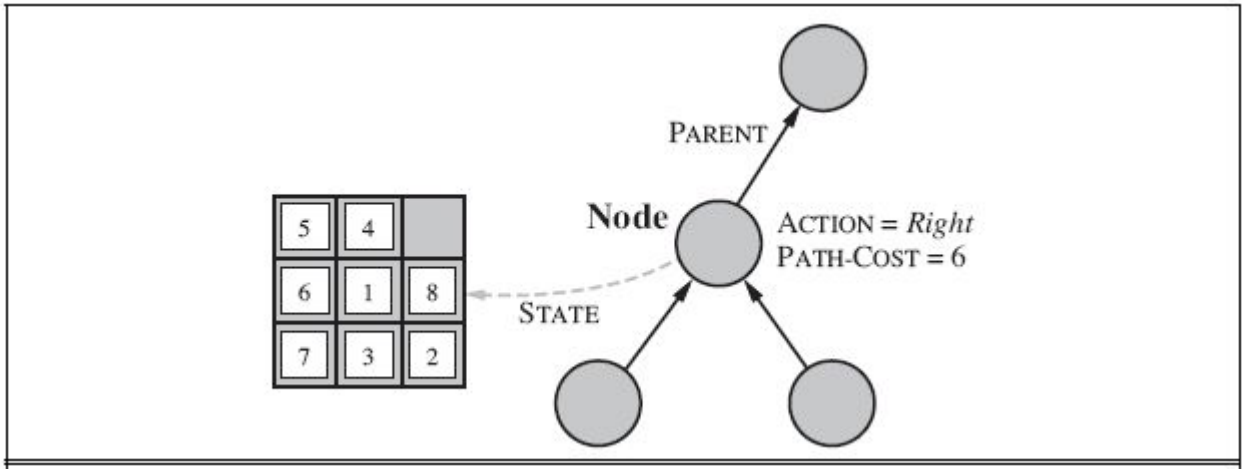
- Search Tree
- Search node
- Expanding
- Generating
- Search Strategy

Node with four components

- STATE
- PARENT-NODE
- ACTION
- PATH-COST


(a) The initial state
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(b) After expanding Arad
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea
(c) After expanding Sibiu
Arad
Sibiu
Timisoara
Zerind
Arad
Fagaras
Oradea
Rimnicu Vilcea
Arad
Lugoj
Arad
Oradea

```
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
```


PARENT
Node
ACTION = Right
PATH-COST = 6
STATE
5
4
6
1
8
7
3
2

Measuring problem-solving performance

- Completeness: Is the algorithm guaranteed to find a solution when there is one?
- Optimality: Does the strategy find the optimal solution?
- Time complexity: How long does it take to find a solution?
- Space complexity: How much memory is needed to perform the search?

End of chapter 3

End of unit-1