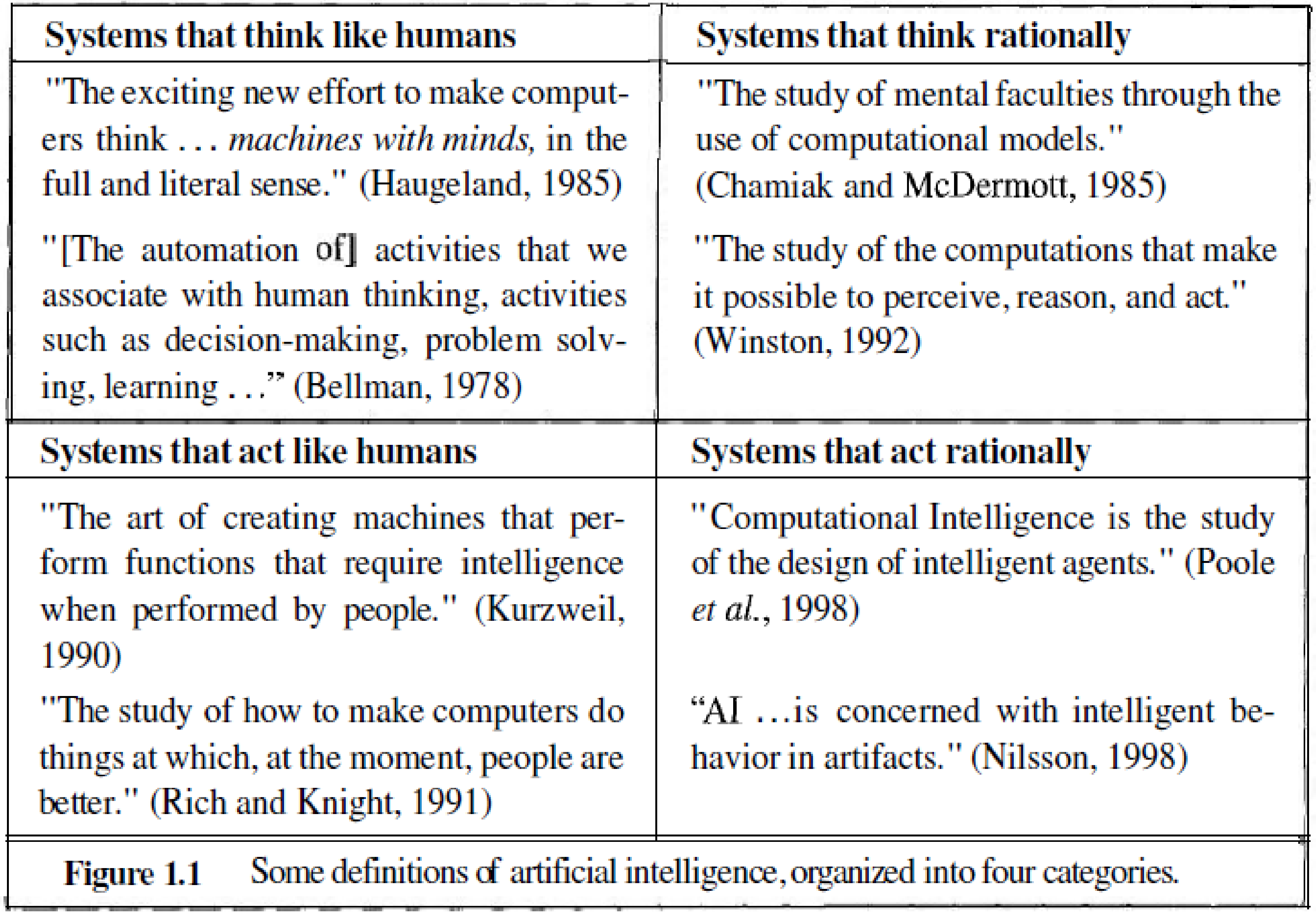
# Agenda

Objectives:

- What is AI

## WHAT is AI?

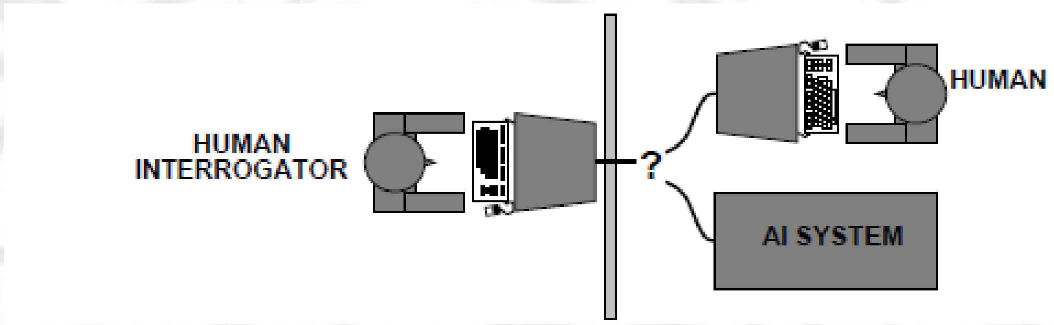


| **Systems that think like humans** | **Systems that think rationally** |
|---|---|
| "The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) | "The study of mental faculties through the use of computational models." (Chamiak and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

**Figure 1.1** Some definitions of artificial intelligence, organized into four categories.

## Acting humanly: The Turing Test approach:

- The Turing Test-provide a satisfactory operational definition of intelligence



- would need to possess
  - **natural language processing**
  - **knowledge representation**
  - **automated reasoning**
  - **machine learning**
- will need **computer vision** and **robotics**
- issue of acting like a human comes up primarily when AI programs have to **interact with people**
- Problem: Turing test is **not reproducible, constructive**, or amenable to mathematical analysis

## Thinking humanly: The cognitive modelling approach:

- The way of determining how humans think.
- need to get inside the actual workings of human minds
  1. **through introspection—trying to catch our own thoughts as they go by**
  2. **through psychological experiments**
- sufficiently precise theory of the mind, possible to express the theory as a computer program.
- Requires scientific theories of internal activities of the brain
- What level of abstraction? \Knowledge" or \circuits"?
- How to validate? Requires
- 1) Predicting and testing behavior of human subjects (top-down): **Cognitive Science**
- or 2) Direct identification from neurological data (bottom-up) : **Cognitive Neuroscience**
- Both share with AI
  - **the available theories do not explain (or engender) anything resembling human-level general intelligence**

## Thinking rationally: The laws of thought approach

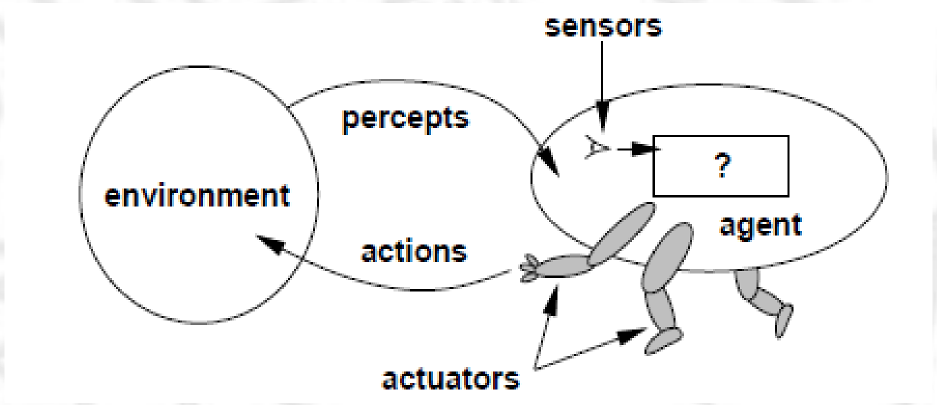
- Normative (or prescriptive) rather than descriptive
- Aristotle: first to attempt to codify "right thinking"
- what are correct arguments/thought processes?
- Several Greek schools developed various forms of logic:
- notation and rules of derivation for thoughts
  - **For example,** "Socrates is a man; all men are mortal; therefore Socrates is mortal."
- Direct line through mathematics and philosophy to modern AI
- Problems:
  - **Not all intelligent behavior is mediated by logical deliberation**
  - **there is a big difference between being able to solve a problem "in principle" and doing so in practice**

## Acting rationally: The rational agent approach

- means acting to achieve one's goals, given one's beliefs.
- The right thing: that which is expected to maximize goal achievement, given the available information
- we need the
  - ability to represent knowledge and reason with it
  - ability to generate comprehensible sentences in natural language
  - learning
  - visual perception
- AI is viewed as the study and construction of rational agents.
- The study of AI as rational agent design therefore has two advantages.
- it is more general than the "laws of thought" approach (correct inference is only a useful mechanism for achieving rationality)
- it is more amenable to scientific development

## WHAT is Agent?

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

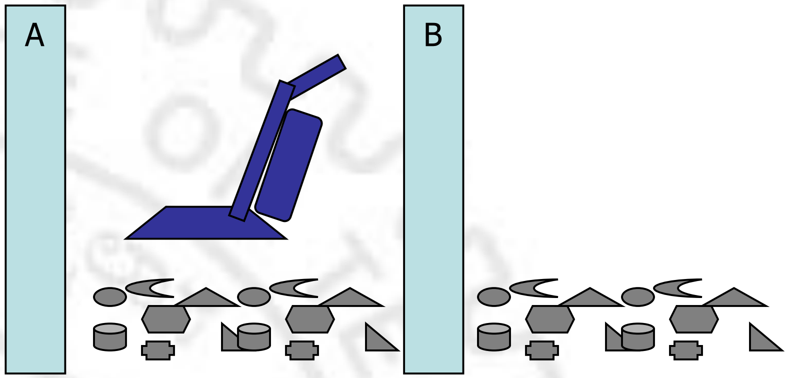


- A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators
- Percept – the agent's perceptual inputs percept sequence is a sequence of everything the agent has ever perceived
- Agent Function – describes the agent's behavior
  - Maps any given percept sequence to an action
  - f : P* -> A
- Agent Program – an implementation of an agent function for an artificial agent

# WHAT is Agent?

## Example: Vacuum Cleaner World

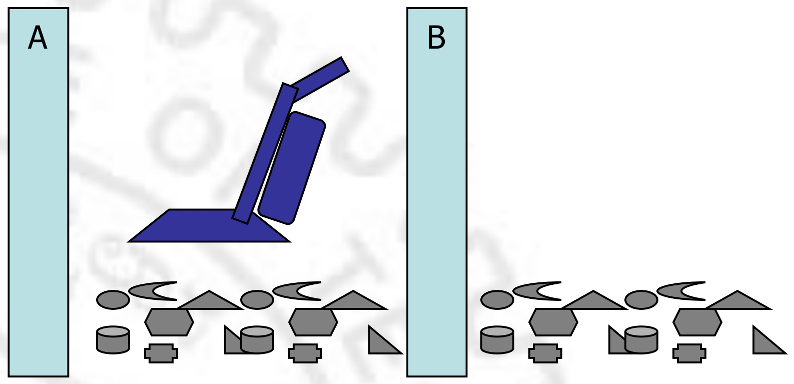
- Two locations: squares A and B
- Perceives what square it is in
- Perceives if there is dirt in the current square
- Actions
  - move left
  - move right
  - suck up the dirt
  - do nothing
- Agent Function
  - If the current square is dirty, then suck, otherwise move to the other square
  - A partial tabulation of this agent function may be like



| 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 |

## WHAT is Agent?

Example: Vacuum Cleaner World



- But what is the right way to fill out the table?
- is the agent
  - good or bad?
  - intelligent or stupid?
- can it be implemented in a small program?

```
Function Reflex-Vacuum-Agent([location, status]) return an action
if status == Dirty then return Suck
else if location = A then return Right
else if location = B then return Left
```

## Good Behavior and Rationality:

- **Rational Agent** – an agent that does the **"right" thing**
  - Every entry in the table is filled out correctly
  - Right thing?
    - the right action is the one that will cause the agent to be most successful
    - need some way to measure success
    - Performance Measure- A scoring function for evaluating the environment space
    - selection of a performance measure is not always easy
- **Rationality:** 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.

- *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

## Omniscience, learning, and autonomy:

- Rationality is not the Omniscience
  - An omniscient agent knows the actual outcome of its actions and can act accordingly
  - rationality does not require omniscience
- Rationality is not the same as perfection
  - Rationality maximizes expected performance, while perfection maximizes actual performance
- Rational agent not only to gather information, but also to learn as much as possible from what it perceives
- A rational agent should be autonomous: does more computation to decide how to modify its behavior
- The behavior of a rational agent can become effectively independent of its prior knowledge

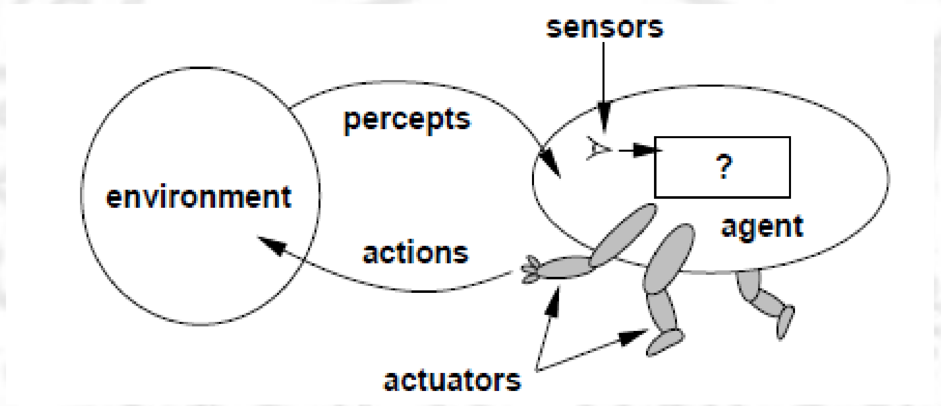
## Agenda

### Objectives:

- Specifying the task environment
- Properties of task environments

# The Nature of Environments:

- *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 the percept sequence and whatever built-in knowledge the agent* has



- The performance measure, the environment, and the agent's actuators and sensors grouping all  together is called the task environment
- Task environments are essentially the "problems" to which rational agents are the "solutions."
- Task environments come in a variety of flavors. The flavor of the task environment directly affects the appropriate design for the agent program.
- PEAS (Performance, Environment, Actuators, Sensors) description

## The Nature of Environments:

### Specifying the task environment:

- “PEAS" is an acronym often used to represent different aspects or components of an intelligent agent
- **Performance Measure (P)**: This refers to the criteria used to evaluate the success of an intelligent agent in achieving its goals. It defines what the agent is supposed to do and how well it should do it.
- **Environment (E)**: This represents the context in which the intelligent agent operates, including the external factors, inputs, and outputs that influence its behavior. The environment could be physical, virtual, or conceptual, depending on the application domain.
- **Actuators (A)**: Actuators are the mechanisms through which an intelligent agent can interact with its environment. These could include physical devices such as motors or sensors, software interfaces, or any other means by which the agent can affect its surroundings.
- **Sensors (S)**: Sensors are the means through which an intelligent agent perceives its environment. These could include cameras, microphones, touch sensors, or any other devices or algorithms that provide input to the agent about its surroundings.

## The Nature of Environments:

**Specifying the task environment:**

Example : A simple vacuum cleaning robot and how we can apply the PEAS framework to it:

**1.Performance Measure (P)**: The percentage of dust and debris removed from the floor within a certain time frame. For instance, the goal might be to achieve at least 95% cleanliness in a room within 30 minutes of operation.

**2.Environment (E)**: The room or space it operates in, which includes various surfaces like hardwood floors, carpets, and tiles. It also includes obstacles like furniture and walls, as well as potential hazards like stairs.

**3.Actuators (A)**: It include the motors that drive its wheels for movement, the vacuum motor that sucks in dirt and debris, and any mechanisms for avoiding obstacles such as bumpers or infrared sensors.

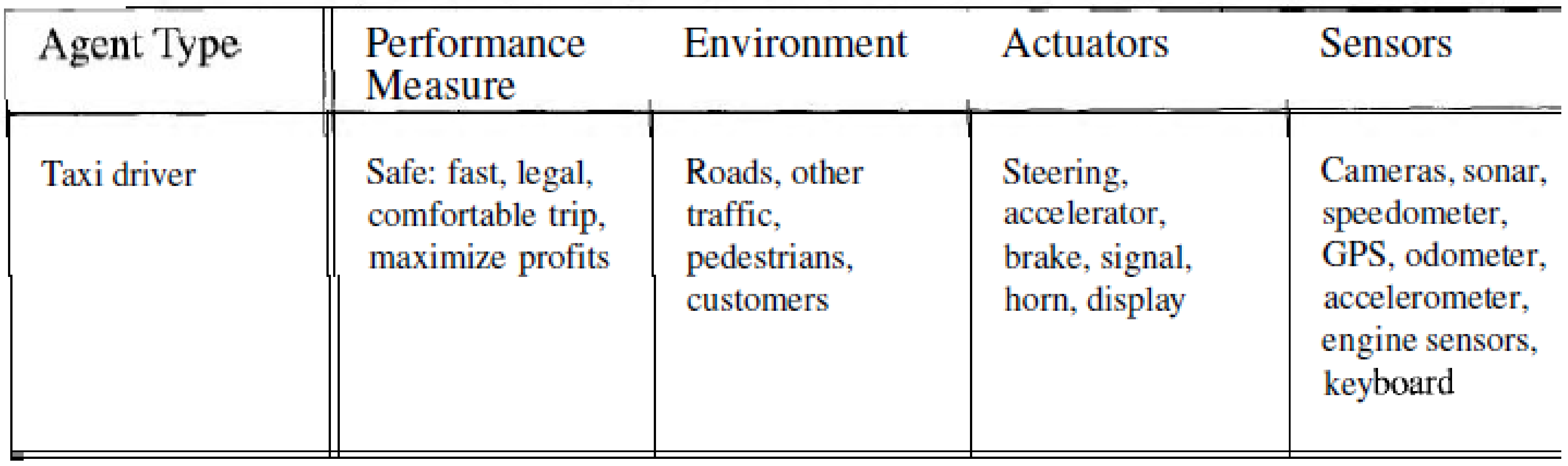
**4.Sensors (S)**: Sensors for the vacuum cleaning robot could include:

1. Infrared or ultrasonic sensors to detect obstacles and avoid collisions.
2. Touch sensors to detect physical contact with obstacles or walls.
3. Optical sensors or cameras to detect dirt and debris on the floor.
4. Wheel encoders to measure distance traveled and navigate the environment

## The Nature of Environments:

**Specifying the task environment:**

- Example : an automated taxi driver.
- Fully automated taxi is currently somewhat beyond the capabilities of existing technology



| 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 |

**Figure 2.4** PEAS description of the task environment for an automated taxi.

## The Nature of Environments:

PEAS description**:**

- **Performance measure**-Desirable qualities include getting to the correct destination; minimizing fuel consumption and wear and tear; minimizing the trip time cost; minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort; maximizing profits
- **Environment:** Deal with a variety of roads, ranging from rural lanes and urban alleys. The roads contain other traffic, pedestrians, stray animals, road works, police cars, puddles, and potholes. The taxi must also interact with potential and actual passengers
- **Actuators**-control over the engine through the accelerator and control over steering and braking
- **Sensors**- One or more controllable TV cameras, the speedometer, and the odometer. Global positioning system (GPS) to give it accurate position with respect to an electronic map, and infrared or sensors to detect distances to other cars and obstacles. Need a keyboard or microphone for the passenger request a destination.

## The Nature of Environments:

Properties of task environments:

- **Fully observable vs. partially observable:**
- Fully observable
  - If an agent's sensors give it access to the complete state of the environment at each point in time
  - if the sensors detect all aspects that are relevant to the choice of action
  - are convenient because the agent need not maintain any internal state to keep track of the world
- partially observable
  - noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data
  - for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking

## The Nature of Environments:

Properties of task environments:

- **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
  - If the environment is partially observable, then it could appear to be stochastic
- **Episodic vs. Sequential:**
  - In an episodic task environment, the agent's experience is divided into atomic episodes.
  - The choice of action in next episode does not depend on the actions taken in previous episodes and it depends only on the episode itself
  - In sequential environments, the current decision could affect all future decisions.
  - Chess and taxi driving are sequential
  - Episodic environments are much simpler than sequential environments

## The Nature of Environments:

Properties of task environments:

- **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
  - Dynamic environments are continuously asking the agent what it wants to do
  - If the environment itself does not change with the passage of time but the agent’s performance score does, then we say the environment is SEMIDYNAMIC
- **Discrete vs. continuous:**
  - Distinction can be applied to the state of the environment, to the way time is handled and to the percepts and actions of the agent
  - A chess game has a finite number of distinct states.
  - Taxi driving is a state and continuous-time problem

## The Nature of Environments:

Properties of task environments:

- **Single agent vs. multiagent:**
  - An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment.
  - In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially cooperative multiagent environment.
  - Chess is a competitive multiagent environment, trying to maximize its performance measure and minimizes opponent performance measure
- As one might expect, the **hardest case** is partially observable, stochastic, sequential, dynamic, continuous, and multiagent.
- The properties are not always cut and dried, depend on how the task environment is defined
- Many environments are episodic at higher levels than the agent's individual actions

## The Nature of Environments:

Properties of task environments: Example

•**Simple Vacuum Cleaning Robot**

1.**Observable vs. Partially Observable**: The environment for a vacuum cleaning robot is typically observable. The robot can use sensors such as cameras or infrared sensors to perceive the layout of the room and detect obstacles and dirt on the floor.

2.**Deterministic vs. Stochastic**: The vacuum cleaning robot operates in a deterministic environment. Its actions, such as moving forward, turning, or activating the vacuum, have predictable outcomes based on the robot's programming and the state of the environment.

3.**Episodic vs. Sequential**: The vacuum cleaning task is sequential. Each action taken by the robot affects the subsequent state of the environment and the progress of the cleaning process until the entire area is cleaned or a predetermined stopping condition is met.

## The Nature of Environments:

Properties of task environments: Example

•**Simple Vacuum Cleaning Robot**

**4.Static vs. Dynamic**: The environment for the vacuum cleaning robot can be considered dynamic. While the layout of the room remains relatively static during cleaning, the presence of people, pets, or objects moving within the environment introduces dynamic elements that the robot must adapt to.

**5.Discrete vs. Continuous**: The actions of the vacuum cleaning robot are typically discrete. The robot can move forward, backward, turn left or right, and activate or deactivate its vacuum suction. These actions are discrete and can be executed in distinct steps.

**6.Single-Agent vs. Multi-Agent**: The vacuum cleaning robot operates as a single agent within its environment. It does not interact with other robots or entities that have their own goals or actions.

## The Nature of Environments:

### Properties of task environments: Example

- **Vending Machine**

**1.Observable vs. Partially Observable**: The state of the vending machine and the items inside it are fully observable to the user. Customers can see the available products, prices, and any instructions or messages displayed on the machine.

**2.Deterministic vs. Stochastic**: The vending machine operates in a deterministic environment. When a customer selects a product and inserts the correct amount of money, the vending machine dispenses the chosen item with certainty, assuming no malfunctions.

**3.Episodic vs. Sequential**: Interactions with a vending machine are episodic. Each transaction is independent of previous transactions. Customers select a product, pay for it, and receive their item without any ongoing sequence of actions or states.

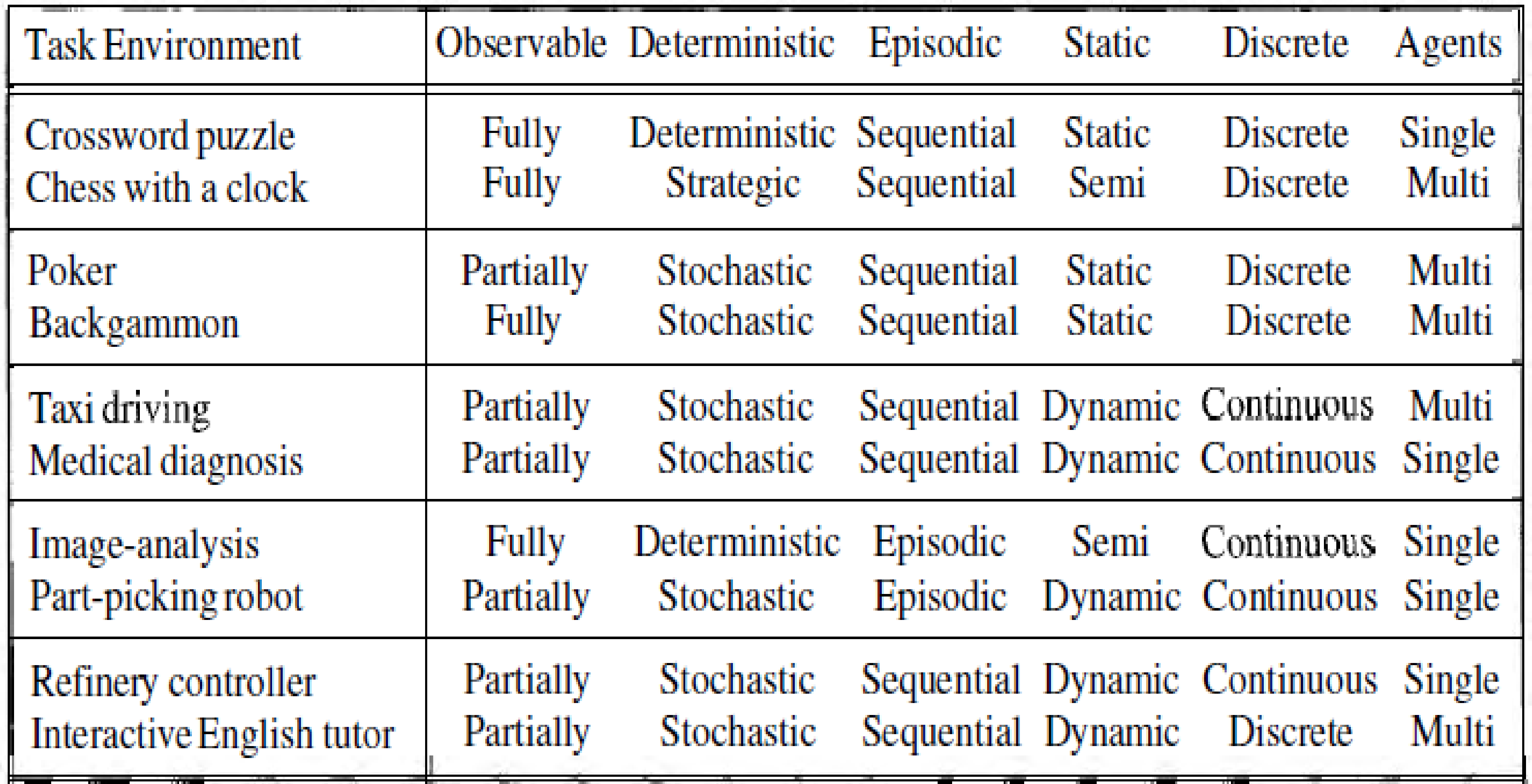
**4.Static vs. Dynamic**: The vending machine environment is generally static. The layout and functionality of the machine remain constant during operation, with no external factors affecting its behavior unless manually modified or experiencing technical issues.

**5.Discrete vs. Continuous**: Actions within the vending machine environment are discrete. Customers make discrete selections from a predefined set of available products, and the machine dispenses items in whole units (e.g., one can of soda, one candy bar).

**6.Single-Agent vs. Multi-Agent**: The vending machine operates as a single agent. It interacts with multiple users (customers) who initiate transactions independently, but the vending machine itself does not engage with other entities that have their own goals or actions.

## The Nature of Environments:

Properties of task environments:



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

**Figure 2.6** Examples of task environments and their characteristics.

## The Nature of Environments:

Properties of task environments:

- For each of the following activities, give a PEAS description of the task environment and characterize it in terms of the properties
  - – Playing soccer.
  - – Playing a tennis match.
  - – Practicing tennis against a wall.
  - – Performing a high jump.
  - – Knitting a sweater.
  - – Bidding on an item at an auction.

# Agenda

Objectives: **Agent programs**

- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents; and
- Utility-based agents.

## THE STRUCTURE OF AGENTS:

- An **agent program** that implements the agent function—the mapping from percepts to actions.
- This program will run on some sort of computing device with physical sensors and actuators—architecture
- agent = architecture + program
- **Agent programs:**
  - Take the current percept as input from the sensors and return an action to the actuators.
  - if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts
  - The key challenge for AI is to **find out how to write programs** that, to the extent possible, produce rational behavior from a small program rather than from a vast table**.**

## THE STRUCTURE OF AGENTS:

### Challenges:

1. no physical agent in this universe will have the **space to store the table**,
2. the designer would **not have time to create the table**,
3. no agent could **ever learn all the right table entries** from its experience, and
4. even if the environment is simple enough to yield a feasible table size, the designer still has **no guidance about how to fill in the table entries**.

## THE STRUCTURE OF AGENTS:

- **Types of agent programs :**
  - we can outline four basic kinds of agent programs
  - Each kind of agent program **combines particular components** in particular ways to generate actions

**1. Simple reflex agents:**

- select actions on the basis of the current percept, ignoring the rest of the percept history.
- Simple reflex behaviors occur even in more complex environments.

## THE STRUCTURE OF AGENTS:

### 1. Simple reflex agents:

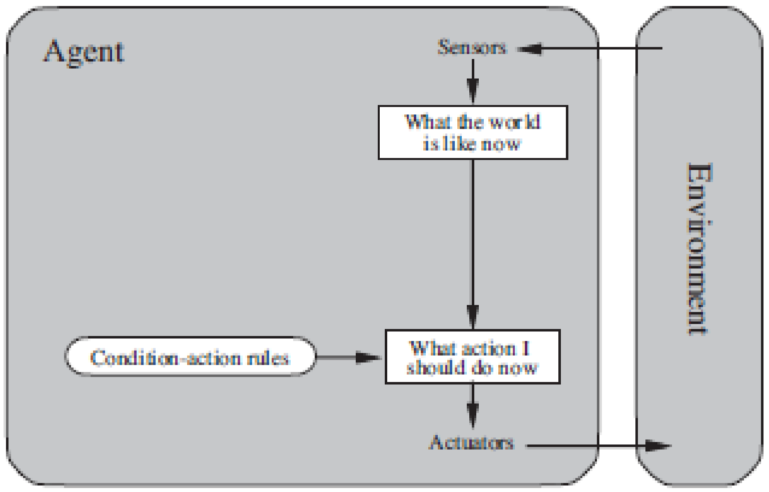
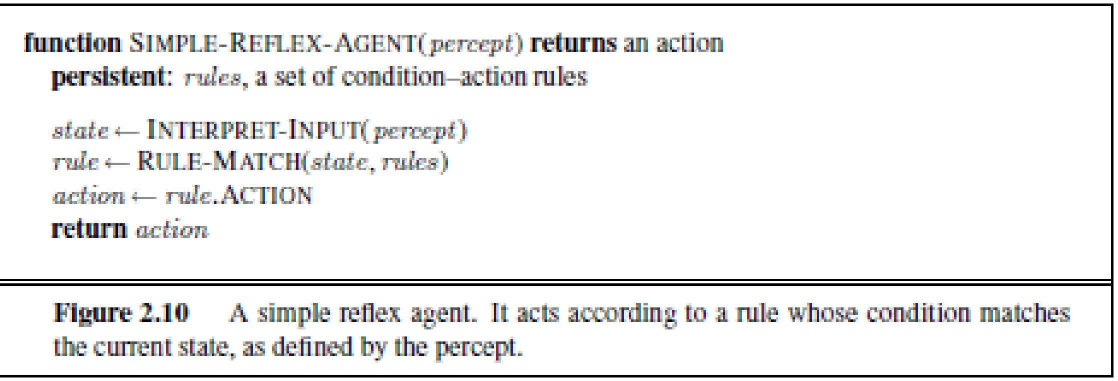


**Figure 2.9** Schematic diagram of a simple reflex agent.



```
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
```

**Figure 2.10** A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

INTERPRET-INPUT function **generates an abstracted description** of the current state from the Percept

RULE-MATCH function **returns the first rule** in the set of rules that matches the given state description

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

Even a little bit of **unobservability** can cause serious trouble.

## THE STRUCTURE OF AGENTS:

### 1. Simple reflex agents: Robotic vacuum cleaner Example

The robot can move left, right, up, or down and can sense whether the square it's currently in is dirty or clean. The goal of the robot is to clean all dirty squares in the room.

The agent's behavior can be described by a simple set of rules:

1. **Perception**: Sense the current state of the environment (whether the current square is dirty or clean).
2. **Action Selection**:

- If the current square is dirty, then the agent performs the "Suck" action to clean the square.
- If the current square is clean, the agent decides its next move based on a predefined sequence of movements. For example, it may decide to move in a particular direction (e.g., left, right, up, or down) until it encounters a dirty square.
- A simple pseudocode representation:

```
while True:
    percept = sense_current_square()

    if percept == "Dirty":
        clean_current_square()
    else:
        move_to_next_square()  # Follow a predefined sequence of movements
```

## THE STRUCTURE OF AGENTS:

### 2. Model-based reflex agents:

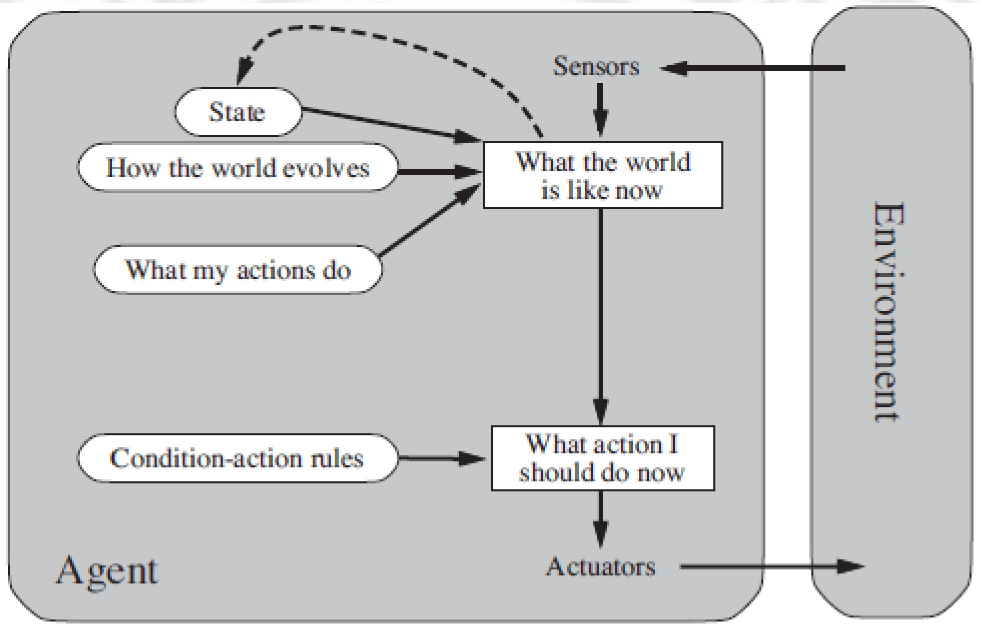
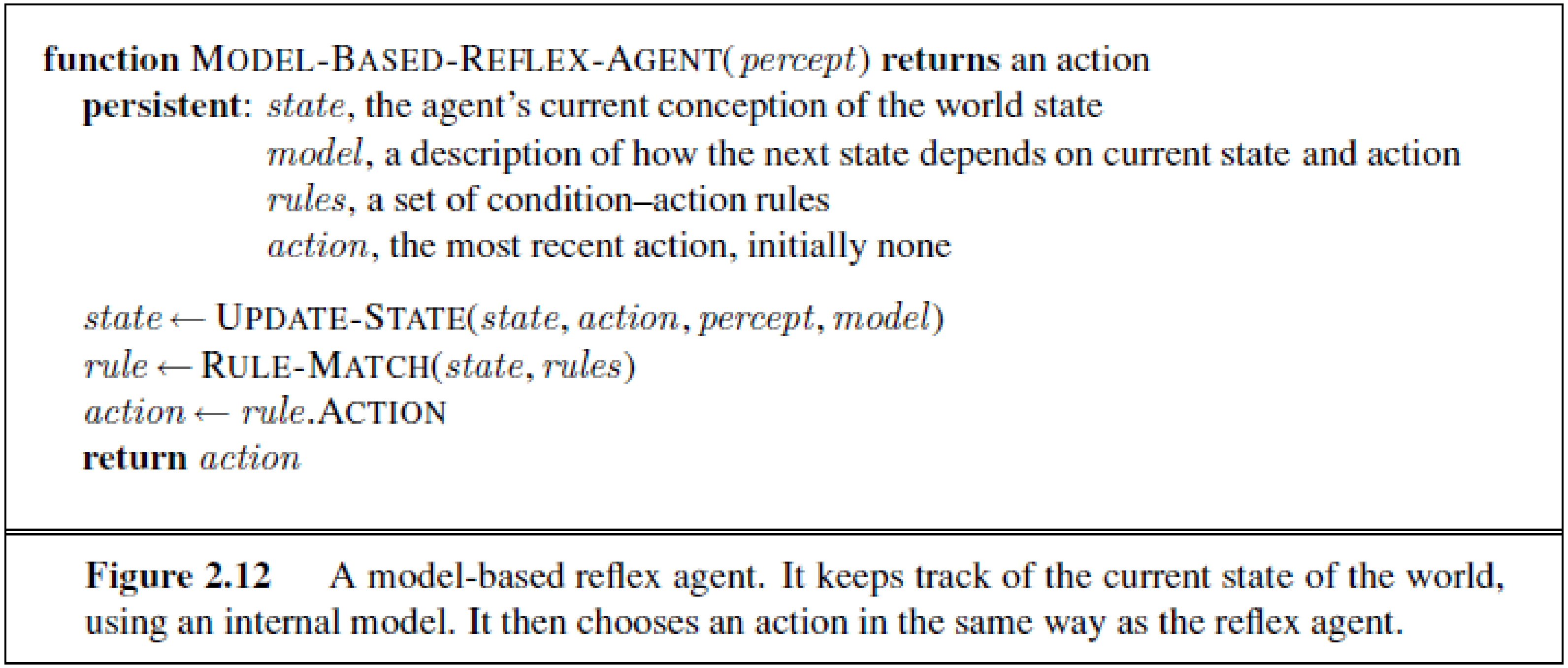


**Figure 2.11** A model-based reflex agent.

- The agent should **maintain some sort of internal state** and thereby reflects at least some of the unobserved aspects of the current state
- Updating this internal state information as time goes by requires **two kinds of knowledge to be encoded** in the agent program
- 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**

## THE STRUCTURE OF AGENTS:

### 2. Model-based reflex 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
```

**Figure 2.12** A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

- The function UPDATE-STATE is responsible for **creating the new internal state description**
- It is seldom possible for the agent to determine the current state of a partially observable environment exactly.

## THE STRUCTURE OF AGENTS:

### 2. Model-based reflex agents: Robotic vacuum cleaner Example

Imagine a robotic vacuum cleaner operating in a room with obstacles and different types of surfaces, such as carpeted areas and hardwood floors. The robot's goal is to efficiently clean the entire room while avoiding obstacles and navigating around furniture.

1. **Perception**: The robot uses sensors to detect the layout of the room, including obstacles and dirty areas. It also senses its current location and the type of surface it's on.
2. **Update Internal Model**: Based on the percept, the robot updates its internal map of the room, marking locations of obstacles and areas that require cleaning.
3. **Decision Making**:
   - If the current location is dirty, the robot decides to clean that spot.
   - If the current location is clean, the robot decides on the next best move based on its internal map and cleaning strategy. This could involve navigating to the nearest dirty spot or following a predefined path to cover the room efficiently while avoiding obstacles.
   - The robot may also need to consider battery levels and recharge if necessary, adjusting its cleaning strategy accordingly

## THE STRUCTURE OF AGENTS:

**2. Model-based reflex agents: Robotic vacuum cleaner Example**

A simplified pseudocode representation:

```
while not all_areas_cleaned:
    percept = sense_environment()
    update_internal_model(percept)

    if current_location_is_dirty():
        clean_current_location()
        continue

    action = select_best_action()
    execute_action(action)
```

In this scenario, the robotic vacuum cleaner acts as a model-based reflex agent by maintaining an internal model of the room and making decisions based on both current percepts and the internal model.

## THE STRUCTURE OF AGENTS:

**3. Goal-based agents:**

– the current state of the environment is not always enough

– the agent needs some sort of goal information

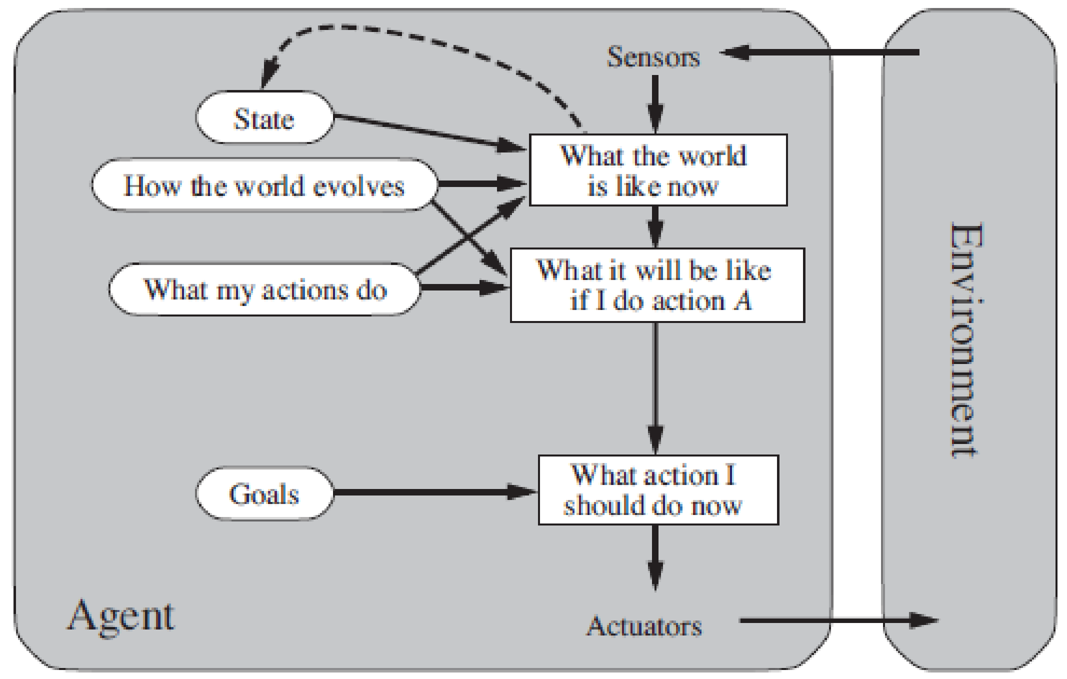

Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action *A*
Goals
What action I should do now
Agent
Actuators
Environment

**Figure 2.13** A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

## THE STRUCTURE OF AGENTS:

### 3. Goal-based agents:

- Sometimes goal-based action selection is straightforward - when goal satisfaction results immediately from a single action.
- Sometimes it will be more tricky - when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.
- Less efficient and more flexible because the knowledge that supports its decisions is represented explicitly and can be modified
- Agent's behavior can easily be changed to go to a different destination, simply by specifying that destination as the goal.
- Goals alone are not enough to generate high-quality behavior in most environments.

## THE STRUCTURE OF AGENTS:

### 3. Goal-based agents: Robotic vacuum cleaner Example

•The agent is driven by explicit goals rather than just reactive responses to its environment.

1. **Goal Formulation**: The agent needs to have a clear goal or goals. The goal could be to clean the entire room thoroughly while minimizing the time taken.
2. **Perception**: The vacuum cleaner's sensors detect the current state of the environment, including dirty areas, obstacles, and the layout of the room.
3. **Knowledge Base**: The agent has knowledge about the layout of the room, including the locations of obstacles, charging stations, and areas that require cleaning.
4. **Decision Making**: Based on its goals and current perception, the agent makes decisions to achieve its goals. This involves planning and reasoning about the best sequence of actions to take.
5. **Action Execution**: The agent executes the planned actions, which may involve moving to specific locations, cleaning dirty areas, avoiding obstacles, and returning to the charging station when necessary.

## THE STRUCTURE OF AGENTS:

### 3. Goal-based agents: Robotic vacuum cleaner Example

- A simplified pseudocode representation:

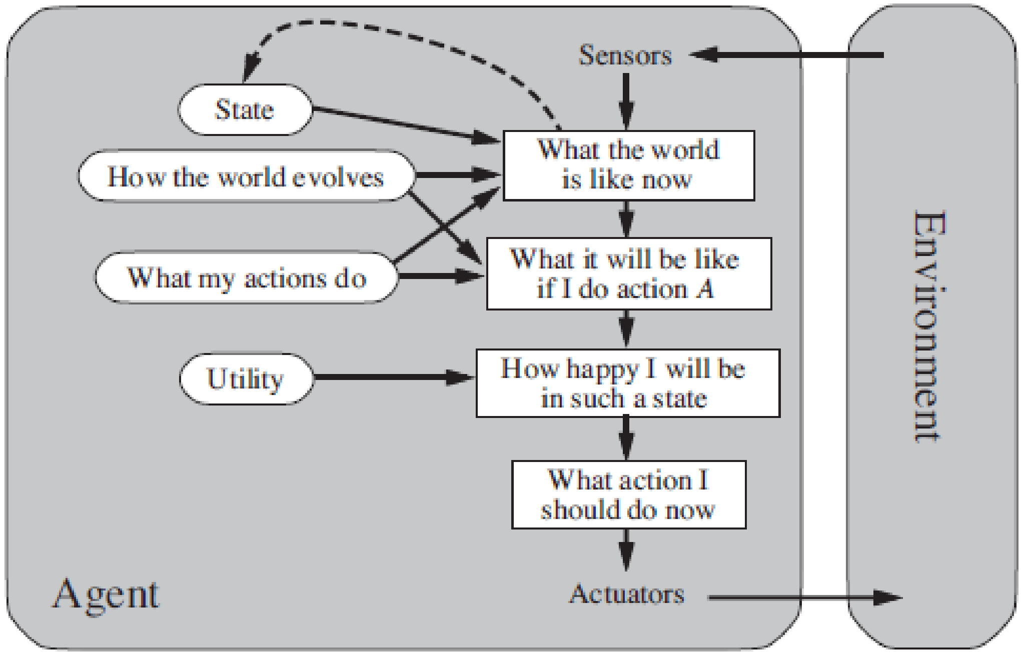
```
while not all_areas_cleaned:
    percept = sense_environment()
        if current_location_is_dirty():
        clean_current_location()
        continue
        if obstacles_in_path():
        plan_path_around_obstacle()
        if low_battery():
        find_nearest_charging_station()
        recharge()
        if all_areas_cleaned():
        return_to_base()
```

- The cleaner is driven by its goal of cleaning the entire room thoroughly. It uses its perception to determine the current state of the environment and makes decisions accordingly to achieve its goal. This might involve cleaning dirty areas, planning paths around obstacles, recharging when necessary, and returning to the charging station once the cleaning task is complete.

## THE STRUCTURE OF AGENTS:

### 4. Utility-based agents:

- 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
- The agent expects to derive, on average, given the probabilities and utilities of each outcome.


Sensors
State
What the world is like now
How the world evolves
What it will be like if I do action A
What my actions do
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

## THE STRUCTURE OF AGENTS:

### 4. Utility-based agents:

- A utility-based agent has many advantages in terms of flexibility and learning
- In two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions.
- First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.
- Second, 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.

## THE STRUCTURE OF AGENTS:

### 4. Utility-based agents: Robotic vacuum cleaner Example

•the agent makes decisions based on maximizing a utility function that evaluates the desirability of different states or outcomes.

•The utility function quantifies how much the agent values different states or outcomes, allowing it to make decisions that maximize its overall utility.

1.**Perception**: The robotic vacuum cleaner's sensors detect the current state of the environment, including dirty areas, obstacles, and the battery level.

2.**Utility Function**: The agent has a utility function that assigns a numerical value to different states or outcomes based on their desirability. For example, clean areas might have a high utility, while dirty areas or obstacles might have a low utility. The utility function also takes into account other factors such as battery level and cleaning time.

3.**Decision Making**: The agent evaluates the expected utility of different actions based on its current perception and knowledge of the environment. It selects the action that maximizes its expected utility. This may involve cleaning dirty areas, avoiding obstacles, recharging the battery, and optimizing cleaning efficiency.

4.**Action Execution**: The agent executes the selected action, updating its perception of the environment and utility function accordingly.

## THE STRUCTURE OF AGENTS:

### 4. Utility-based agents: Robotic vacuum cleaner Example

- a simplified pseudocode representation:

```
• while not all_areas_cleaned:
•     percept = sense_environment()
•     calculate_expected_utilities(percept)
•   select_action_with_maximum_expected_utility()
•     execute_selected_action()
```

- utility-based approach allows the vacuum cleaner to make decisions that maximize its utility, considering factors such as cleaning efficiency and battery consumption

## Agenda

Objectives:

**Learning agents**

**PROBLEM-SOLVING AGENTS**

## THE STRUCTURE OF AGENTS:

- **Learning agents:**
- How the agent programs come into being?
- A learning agent can be divided into four conceptual components

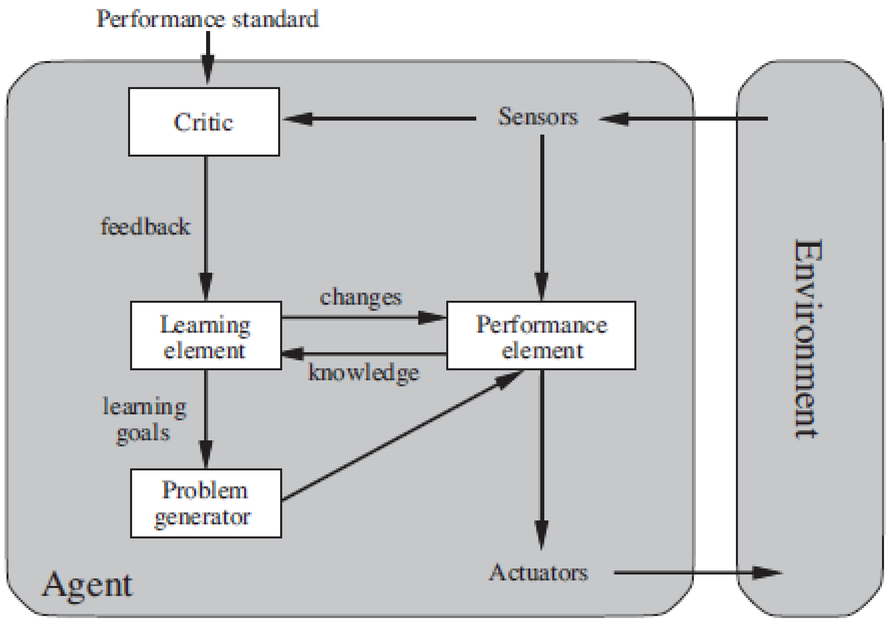


**Figure 2.15** A general learning agent.

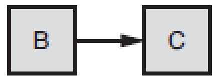
**Learning element**, responsible for making improvements
The learning element can make changes to any of the "knowledge" components
**Performance element** responsible for selecting external actions. it takes in percepts and decides on actions
The **critic** tells the learning element how well the agent is doing with respect to a fixed performance standard
**Problem generator** is responsible for suggesting actions that will lead to new and informative experiences

## THE STRUCTURE OF AGENTS:

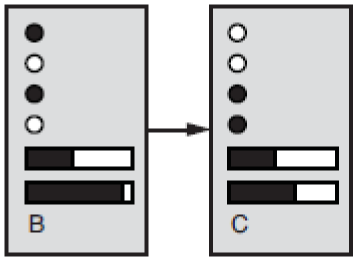
- **Learning agents:**
- if the agent is willing to explore a little and discover much better actions for the long run
- The problem generator might identify certain areas of behavior in need of improvement and suggest experiments
- The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.
- A process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent

## THE STRUCTURE OF AGENTS:

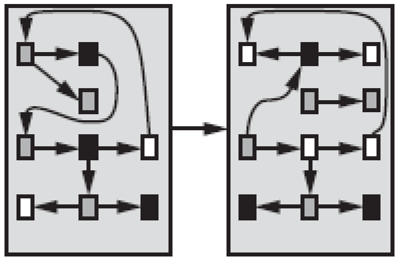
**How the components of agent programs work:**



(a) Atomic



(b) Factored



(b) Structured

- We can place the representations along an axis of increasing complexity and expressive power
- In an **atomic** representation each state of the world is indivisible—it has no internal structure.
- **factored** representation splits up each state into a fixed set of variables or attributes, each of which can have a value
  - Two different factored states can share some attributes
  - Much easier to work out how to turn one state into another. With factored representations, we can also represent uncertainty
  - Many important areas of AI are based on factored representations,
  - constraint satisfaction algorithms, propositional logic, planning, Bayesian networks and the machine learning algorithms
  - For many purposes, we need to understand the world as having things in it that are related to each other, not just variables with values
- **Structured** representations underlie relational databases and first-order logic, first-order probability models, knowledge-based learning and much of natural language understanding

## Simple problem solving agent:

- The  simple reflex agents, which base their actions on a direct mapping from states to actions. Such agents cannot operate well in environments
- Goal-based agents, consider future actions and the desirability of their outcomes
- One kind of goal-based agent called **a problem-solving agent,** with no internal structure visible to the problem solving algorithms.
- **Uninformed search algorithms**-given no information about the problem other than its definition
- **Informed search algorithms**-given some guidance on where to look for solutions.

## PROBLEM-SOLVING AGENTS:

- adopt a **goal** and aim at satisfying it.
- The agent's **performance measure** contains many factors
- The decision problem is a **complex** one involving many tradeoffs
- **Goals** help organize behaviour by limiting the objectives that the agent is trying to achieve
- **Goal formulation**, is the first step in problem solving, based on the current situation and the agent's performance measure.
- We will consider a goal to be a **set of world states.**
- The agent's task is to find out how to act, now and in the future, so that it reaches a goal state.
  - i.e. it needs to decide what sorts of actions and states it should consider
  - too much uncertainty -- too many steps in a solution.

## PROBLEM-SOLVING AGENTS:

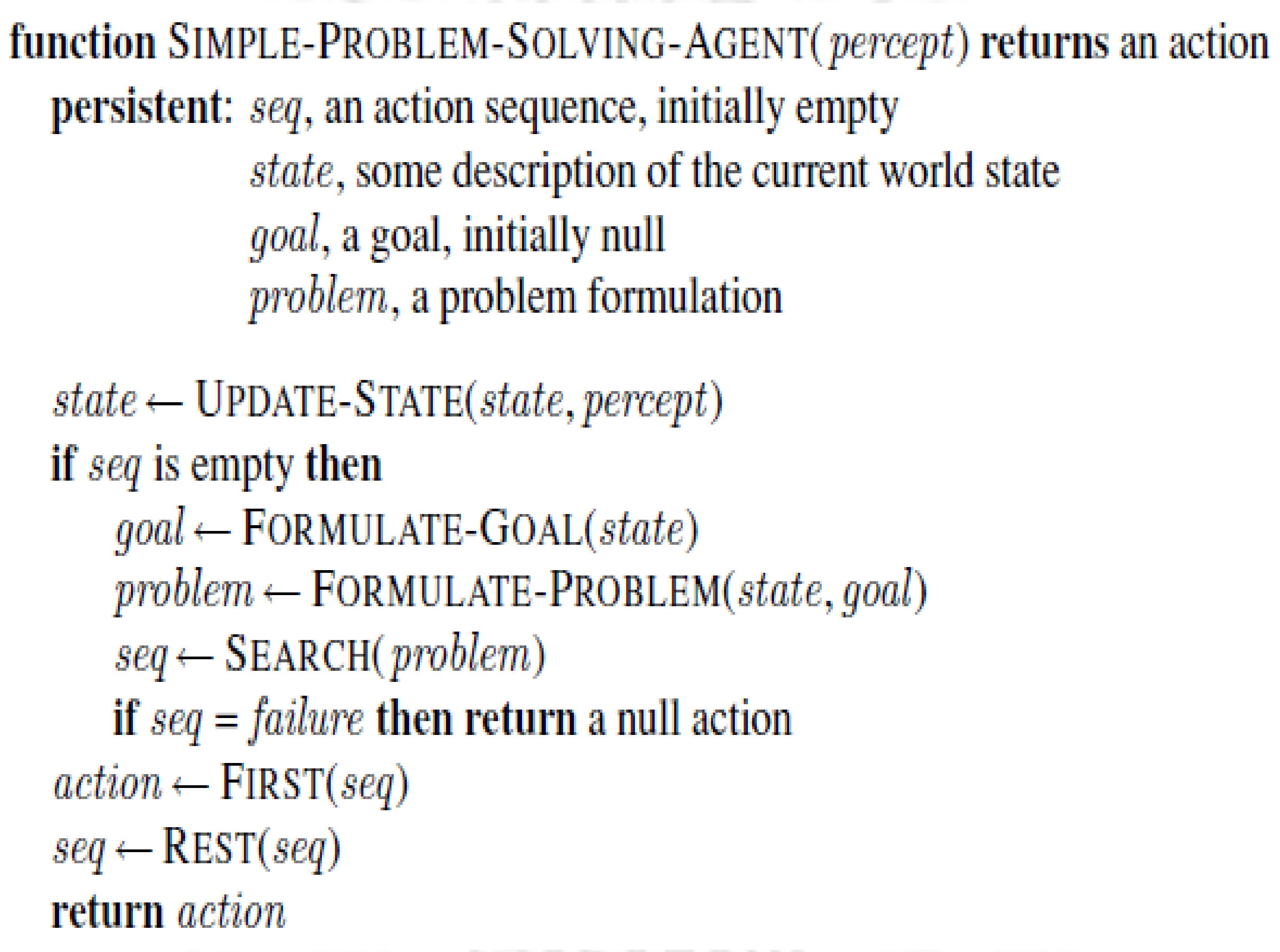
- **Problem formulation** is the process of deciding what actions and states to consider, given a goal.
- if the environment is
  - **unknown**, then it is has no choice but to try random.
  - The agent can use available information to consider subsequent stages
- In general, an agent with **several immediate options** of unknown value can decide what to do by first examining future actions that eventually lead to states of known value.
- To **examining future actions** we have to be more specific about properties of the environment,
- we assume that the environment is
  - **observable**, so the agent always knows the current state
  - **discrete**, so at any given state there are only finitely many
  - **known**, so the agent knows which states are reached by each action.
  - **deterministic**, so each action has exactly one outcome.

Under these assumptions, the solution to any problem is a fixed sequence of actions

## PROBLEM-SOLVING AGENTS:

- In general it could be a branching strategy that recommends different actions in the future depending on what percepts arrive
- The process of looking for a sequence of actions that reaches the goal is called **search**.
- A search algorithm takes a problem as input and returns a solution in the form of an **action sequence**.
- Once a solution is found, the actions it recommends can be carried out. This is called the **execution phase**.
- After **formulating a goal** and a **problem to solve**, the agent calls a **search procedure** to solve it. It then uses the solution to **guide its actions**
- Once the solution has been executed, the agent will formulate a new goal

## PROBLEM-SOLVING AGENTS:



```
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-SOLVING AGENTS:

1. **Initialization**: The agent initializes its internal state, including the current world state, goal, and problem formulation.
2. **Percept Processing**: The agent updates its internal state based on the percept received from the environment.
3. **Goal Formulation**: If the action sequence is empty (meaning the agent doesn't have a plan), the agent formulates its goal based on the current world state.
4. **Problem Formulation**: The agent formulates a problem based on the current world state and the formulated goal.
5. **Search**: The agent searches for a solution sequence using a search algorithm (e.g., depth-first search, breadth-first search) applied to the formulated problem.
6. **Action Execution**: If a solution sequence is found, the agent takes the first action from the sequence and updates the sequence by removing the executed action.
7. **Return Action**: The agent returns the selected action to be executed in the environment.

**PROBLEM-SOLVING AGENTS:** searching a binary tree using BFS,

1. Initialize the frontier with the root node of the binary tree.
2. While the frontier is not empty:
    a. Dequeue a node from the frontier.
    b. Check if the dequeued node satisfies the goal condition.
    c. If the goal is reached, return the solution.
    d. If the goal is not reached, enqueue the children of the dequeued node into the frontier.
3. If the frontier becomes empty and the goal is not reached, return failure.

(BFS visits the nodes level by level, starting from the root node (1) and visiting its children (2 and 3), then their children (4, 5, 6, and 7) in order.)

Reference: SIMPLE-PROBLEM-SOLVING-AGENT function for searching a binary tree using BFS

## PROBLEM-SOLVING AGENTS: vacuum cleaning robot

Let's assume the robot's percept is (1, 0, [(0, 1)]), indicating that the robot is at position (1, 0) and there is a dirty spot at position (0, 1).

1. **Initialization**:
   seq: Empty
   state: Current world state (with robot's position and dirty spots)
   goal: Null
   problem: Null
2. **Percept Processing**: Update the state based on the received percept:
   state ← UPDATE-STATE(state, (1, 0, [(0, 1)]))
3. **Goal Formulation**: The goal is to clean all dirty spots:
   goal ← FORMULATE-GOAL(state)
4. **Problem Formulation**: Formulate a problem based on the current state and the formulated goal:
   problem ← FORMULATE-PROBLEM(state, goal)
5. **Search:** Search for a solution sequence using a search algorithm:
   seq ← SEARCH(problem)
6. **Action Execution:** If a solution sequence is found, execute the first action from the sequence and update the sequence:
   action ← FIRST(seq) and seq ← REST(seq)
7. **Return Action:** Return the selected action: return action