Environment and Agent

Dr. Sumanto Dutta
School of Computer Applications
Kalinga Institute of Industrial Technology (KIIT)

January 12, 2024

Contents

1. The Concept Of Rationality
   - Rationality
2. The Nature Of Environments
   - Specifying the Task Environment
   - Properties of Task Environments
3. Agent Programming
   - Types of agent programs: Simple Reflex Agents
   - Types of agent programs: Model-based Reflex Agents
   - Types of agent programs: Goal-based Agents
   - Types of agent programs: Utility-based Agents
   - Example Case Study

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.

Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not, having a certain agent function shown in Figure 6. Is this a rational agent?

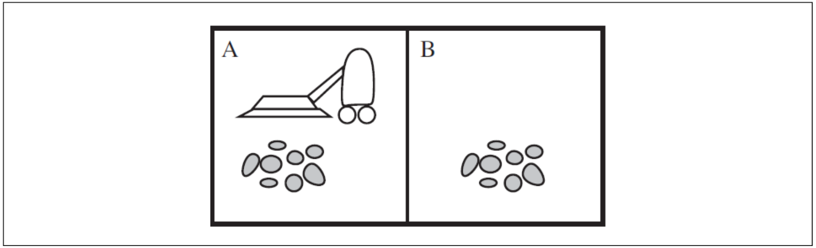


Figure 1: A vacuum-cleaner world with just two locations..[1]

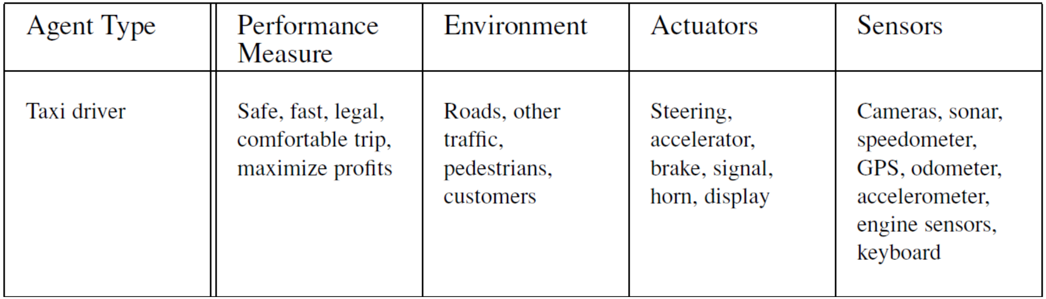
[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

- Let us assume the following :
    - The performance measure awards one point for each clean square at each time step, over a “lifetime” of 1000 time steps.
    - The “geography” of the environment is known a priori but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and cleaning action cleans the current square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
    - The only available actions are Left , Right, and Cleaning.
    - The agent correctly perceives its location and whether that location contains dirt.
- *Under these circumstances, the agent is indeed rational; its expected performance is at least as high as any other agent's.*
- **When vacuum-cleaner agent cleaner agent would be irrational ?**
    - Once all the dirt is cleaned up, the agent will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly.
    - If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B.

Specifying the Task Environment

Task Environment

The **P**erformance measure, the **E**nvironment, and the agent's **A**ctuators and **S**ensors (PEAS) group under the heading of the **task environment**.

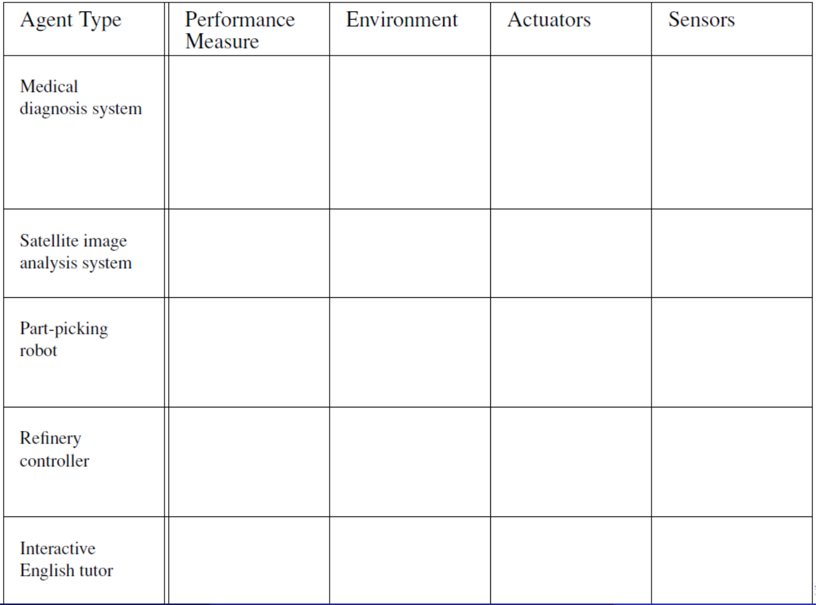


Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure 2: PEAS description of the task environment for an automated taxi.[1]

[1] Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

Let's fill this table shown in Figure 17 together to check how much we understood PEAS.



Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system				
Satellite image analysis system				
Part-picking robot				
Refinery controller				
Interactive English tutor				

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

Properties of Task Environments

Fully 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.
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.

Partially Observable

- A Partially Observable environment in Artificial Intelligence is one where the agent does not have complete information about the current state of the environment.
- The agent can only observe a subset of the environment, and some aspects of the environment may be hidden or uncertain.

Unobservable

- If the agent has no sensors at all then the environment is unobservable.

Single Agent

- a single-agent task environment refers to a scenario where there is only one agent that interacts with an environment to achieve a specific goal.
- The agent takes actions in the environment, and the environment responds to these actions, providing feedback or changing its state.
- Example.

Multi-Agent

- In a multi-agent task environment, there are multiple agents that interact with each other and the environment to achieve individual and/or collective goals.
    - **Competitive** - A competitive multi-agent environment is a scenario in which multiple autonomous agents interact within a shared environment, each pursuing its own objectives and competing against other agents to achieve superior outcomes.
    - **Cooperative** - A cooperative multi-agent environment involves multiple autonomous agents collaborating to achieve shared goals within a common environment. These agents work together, coordinating their actions and strategies to maximize collective performance.
- Example.

Deterministic

- 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.
- Example.

Stochastic

- If the next state of the environment is not completely determined by the current state and the action executed by the agent, then we say the environment is stochastic.
- "Stochastic" generally implies that uncertainty about outcomes is quantified in terms of probabilities
- Example.

Uncertain

- An environment is uncertain if it is not fully observable or not deterministic.
- Example.

A **nondeterministic** *environment is one in which actions are characterized by their possible outcomes, but no probabilities are attached to them.*

Episodic

- In an episodic task environment, the agent's experience is SEQUENTIAL divided into atomic episodes. In each episode the agent receives a percept and then performs a single action.
- The next episode does not depend on the actions taken in the previous episodes.
- Example.

Sequential

- In sequential environments, on the other hand, the current decision could affect all future decisions.
- Example.

Static

- A static environment is one that does not change while an agent is deliberating.
- Example.

Dynamic

- A dynamic environment is one that keeps constantly changing itself when the agent is up with some action.
- Example.

Semi-Dynamic

- A semi-dynamic environment is one where the environment itself does not change with time, but the agent's performance score does.
- Example.

Discrete

- A discrete environment is one in which there are a finite number of percepts and actions that can be performed within it.
- Example.

Continuous

- A continuous environment is one where there are an infinite number of percepts and actions that can be performed within it.
- Example.

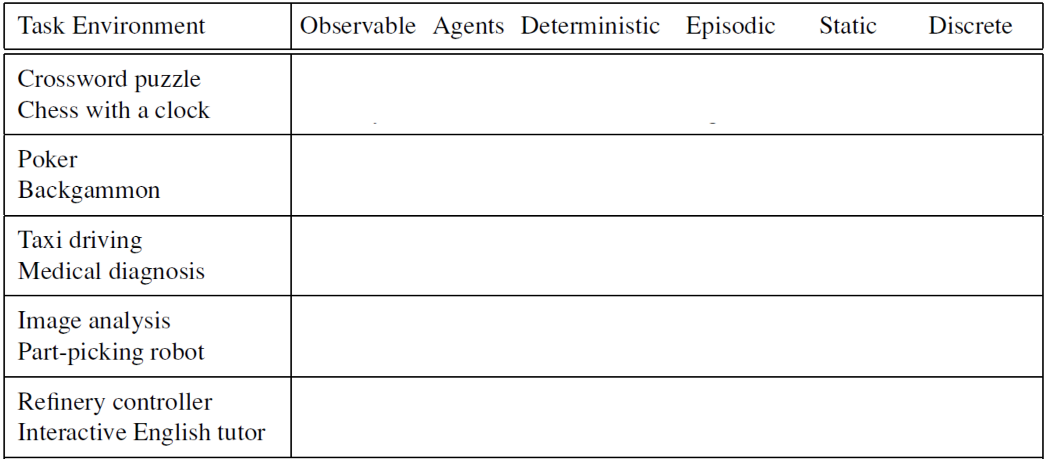
Known

- A known environment is one in which the agent has complete knowledge of the environment's rules, state transitions, and reward structure.
- Example.

Unknown

- An unknown environment is one where the agent does not have complete knowledge of the environment's rules, state transitions, and reward structure.
- Example.

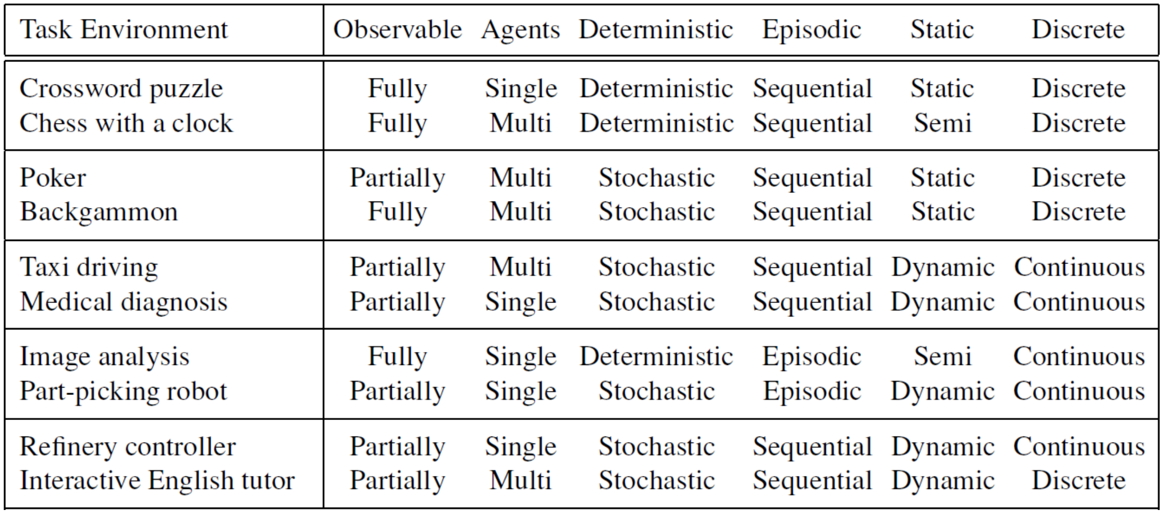
Let's fill this table shown in Figure 17 together to check how much we understood environment.



Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle Chess with a clock						
Poker Backgammon						
Taxi driving Medical diagnosis						
Image analysis Part-picking robot						
Refinery controller Interactive English tutor						

Figure 5: Examples of task environments and their characteristics.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.



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

Figure 6: Examples of task environments and their characteristics.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

Agent Programming

Agent

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

- Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- Robotic agent: cameras and infrared range finders for sensors; various motors for actuators

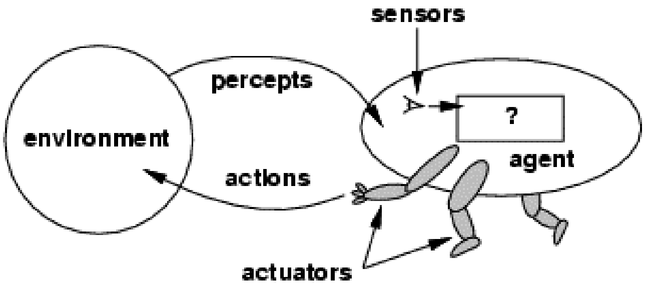


Figure 7: Agent.

Properties

- Autonomous
- Interacts with other agents plus the environment
- Reactive to the environment
- Pro-active (goal-directed)

Environment

The outside circumstances in which an agent functions to accomplish a particular task.

Agent Function

The agent function is a mathematical function that maps a sequence of perceptions into action.

$$[f = P^* \to A]$$

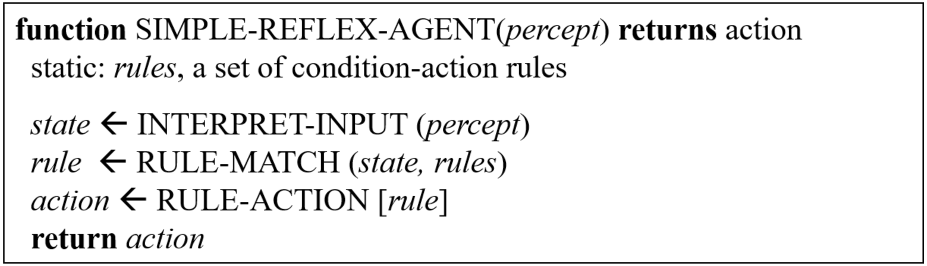
Agent Program

An agent program is an implementation of an agent function. The agent program runs on the physical architecture to produce f.

Agent = Architecture + Program

Simple Reflex Agents

- The simplest kind of agent is the **simple reflex agent**.
- These agents select actions based on the current percept, ignoring the rest of the percept history.
- A simple reflex agent acts according to a condition-action rule whose condition matches the current state, as defined by the percept.
- They are stateless devices that do not have memory of past world states.



```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT (percept)
  rule  ← RULE-MATCH (state, rules)
  action ← RULE-ACTION [rule]
  return action
```

Figure 8: Pseudocode for Simple Reflex Agent.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

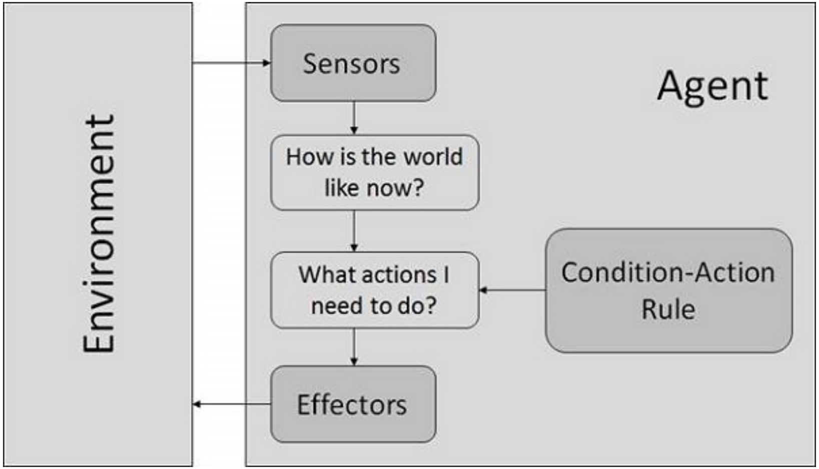


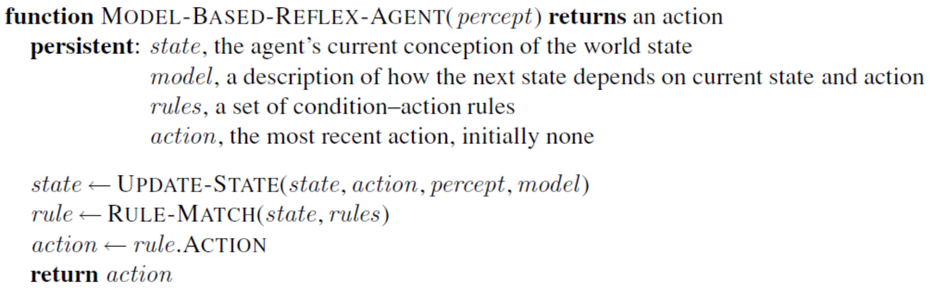
Figure 9: Simple Reflex Agents.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

- Table lookup of condition-action pairs defining all possible condition-action rules necessary to interact in an environment.
- Problems
    - Table is still too big to generate and to store (e.g. taxi)
    - Takes long time to build the table.
    - No knowledge of non-perceptual parts of the current state.
    - Not adaptive to changes in the environment; requires entire table to be updated if changes occur.
    - Looping: Can't make actions conditional.

Model-based Reflex Agents

- Decision-making based on current percept and an internal model of the world.
- Comprehensive representation of the environment. Includes current state, past states, and predictions about future states.
- The advantage of using a model-based approach is that it allows the agent to make more informed decisions by considering not only the immediate environment but also the historical context and potential future states..



```
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

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

Figure 10: Pseudocode for Model-based Reflex Agent.[1]

[1] Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

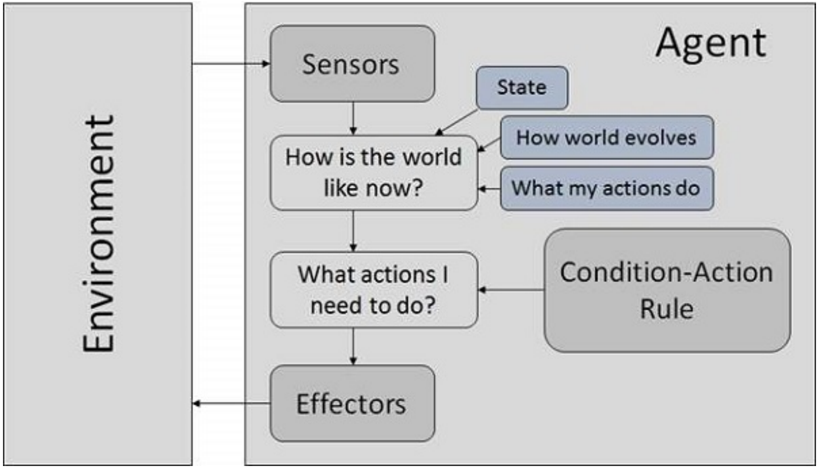


Figure 11: Model-based reflex agents.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

- Model-based reflex agents are commonly used in areas such as robotics, where the agent needs to navigate and interact with a changing environment.
- Problems
  - Building an accurate and comprehensive internal model of the world can be challenging, especially in dynamic and uncertain environments.
  - Maintaining and updating an internal model can be computationally intensive, especially in real-time applications.
  - The constant processing of perceptual data and updating of the internal model may lead to a high cognitive load on the agent.

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 desirable situations.
- The agent program can combine the goal with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal is known as a goal-based agent.
- The goal-based agent is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified.
- A goal-based agent 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 achieving its goals.

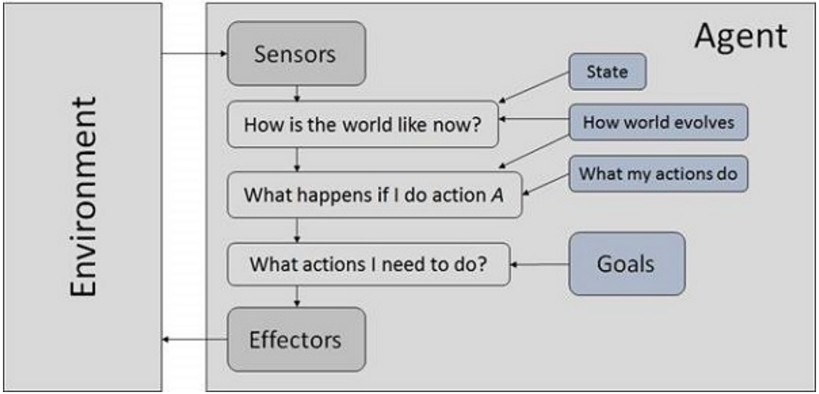

Environment
Agent
Sensors
State
How is the world like now?
How world evolves
What happens if I do action *A*
What my actions do
What actions I need to do?
Goals
Effectors

Figure 12: Goal-based agents.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

- Goal-based agents are employed in various domains, including robotics, artificial intelligence planning, autonomous systems, game playing, and business process automation. Their ability to operate with explicit goals makes them well-suited for complex and dynamic environments where intelligent decision-making is essential.
- Problems
  - The agent's performance heavily relies on the accuracy and completeness of the specified goals.
  - Goals may sometimes conflict with each other, requiring the agent to prioritize and make trade-offs.
  - Planning and decision-making based on goals can be computationally intensive, especially in large and complex problem spaces.
  - In multi-agent systems, coordinating goals with other agents and communicating effectively can be challenging.

Utility-based Agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- Goals just provide a crude binary distinction between “happy” and “unhappy” states.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent.
- An agent’s utility function essentially internalizes the performance measure. If the internal utility function and the external performance measure agree, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.
- Utility-based agents use a model of the world, along with a utility function that measures its preferences among states of the world. Then, it chooses the action that leads to the best-expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome

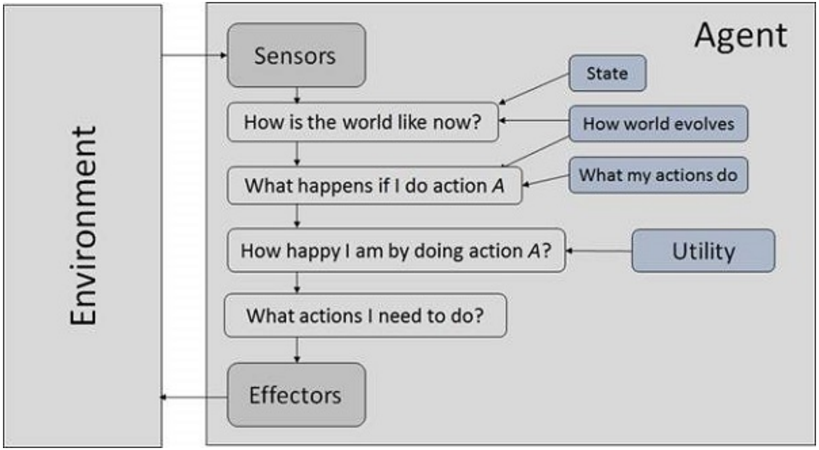

Environment
Agent
Sensors
State
How is the world like now?
How world evolves
What happens if I do action A
What my actions do
How happy I am by doing action A?
Utility
What actions I need to do?
Effectors

Figure 13: Utility-based Agents.[1]

[1]Russell, S. J., Norvig, P. (2021). Artificial Intelligence: A Modern Approach. United Kingdom: Pearson.

- Utility-based agents are applied in various domains, including economics, decision theory, game playing, resource allocation, and optimization problems. Their ability to explicitly consider preferences and make decisions based on maximizing expected utility makes them particularly useful in situations where outcomes are uncertain or involve multiple competing criteria.
- Problems
    - Designing an accurate and comprehensive utility function can be challenging.
    - Utility is inherently subjective and can vary between individuals. Determining an agent's true preferences and translating them into a utility function may involve ambiguity and uncertainty.
    - Evaluating the utility of different outcomes can be computationally intensive, especially when dealing with a large number of possible states or when incorporating complex probability distributions.
    - The emphasis on quantifying preferences in a utility function might oversimplify decision-making. Some aspects of decision problems may be difficult to represent numerically.
    - Utility-based models may struggle with incorporating non-quantifiable factors such as ethics, social norms, or cultural considerations, which play a crucial role in decision-making.

Example Case Study

Problem Statement

Simulate a simple two-room vacuum cleaner world where a vacuum cleaner agent is placed in a house with two rooms (Room A and Room B), and its goal is to clean the dirty rooms.

Environment Description

- The house has two rooms: Room A and Room B.
- Each room can be in one of two states: "Clean" or "Dirty".
- The agent starts in one of the rooms at random.
- The agent can move between rooms and clean the rooms.

Agent Description

- The vacuum cleaner agent uses a Simple Reflex approach to make decisions.
- The agent has the following sensors:
  - Current room location (Room A or Room B).
  - Dirt sensor (detects if the current room is dirty).

Agent's Decision-Making

- The agent's decision-making is based on a set of simple rules:
  - If the current room is dirty, the agent cleans the room.
  - If the current room is clean, the agent moves to the other room.

Tasks

1. Implement the **'SimpleVacuumEnvironment'** class that represents the vacuum cleaner world and includes methods to:
   - Check if a room is dirty.
   - Clean a room.
   - Move the agent to another room.
   - Display the current state of the environment.
2. Implement the **'SimpleReflexVacuumAgent'** class that represents the vacuum cleaner agent using a Simple Reflex approach. The agent should have methods to:
   - Perceive the current room and dirt status.
   - Decide actions based on the rules described above.
   - Perform the decided action.
3. Create an instance for each class and simulate the agent's actions for a few steps, displaying the environment's state after each action.

Problem Statement

Simulate a simple two-room vacuum cleaner world where a vacuum cleaner agent is placed in a house with two rooms (Room A and Room B), and its goal is to clean the dirty rooms.

Environment Description

- The house has two rooms: Room A and Room B.
- Each room can be in one of two states: "Clean" or "Dirty".
- The agent starts in one of the rooms at random.
- The agent can move between rooms and clean the rooms.

Agent Description

- The vacuum cleaner agent uses a Model-Based Reflex approach to make decisions.
- The agent has the following sensors:
  - Current room location (Room A or Room B).
  - Dirt sensor (detects if the current room is dirty).
- The agent maintains a model of the environment, which includes the dirt status of both rooms.

Agent's Decision-Making

- The agent's decision-making is based on following rules:
  - If the current room is dirty based on the actual state and the model, the agent cleans the room.
  - If the current room is clean based on the actual state but dirty in the model, the agent cleans the room based on the model.
  - Otherwise, the agent moves to the other room.

Tasks

1. Implement the **'SimpleVacuumEnvironment'** class that represents the vacuum cleaner world and includes methods to:
   - Check if a room is dirty.
   - Clean a room.
   - Move the agent to another room.
   - Display the current state of the environment.
2. Implement the **'ModelBasedReflexVacuumAgent'** class that represents the vacuum cleaner agent using a Model-Based Reflex approach. The agent should have methods to:
   - Perceive the current room and dirt status.
   - Maintain and update a model of the environment.
   - Decide actions based on the rules described above.
   - Perform the decided action.
3. Create an instance for each class and simulate the agent's actions for a few steps, displaying the environment's state after each action.

Problem Statement

Simulate a simple two-room vacuum cleaner world where a vacuum cleaner agent is placed in a house with two rooms (Room A and Room B), and its goal is to clean the dirty rooms.

Environment Description

- The house has two rooms: Room A and Room B.
- Each room can be in one of two states: "Clean" or "Dirty".
- The agent starts in one of the rooms at random.
- The agent can move between rooms and clean the rooms.

Agent Description

- The vacuum cleaner agent uses a Goal-Based Agent approach to make decisions.
- The agent has the following sensors:
    - Current room location (Room A or Room B).
    - Dirt sensor (detects if the current room is dirty).

Agent's Decision-Making

- The agent's decision-making is based on explicit goals and priorities:
  - The agent maintains a list of goals, each represented as a tuple with an action and a priority.
  - The agent prioritizes and selects the highest-priority goal and performs the associated action.

Tasks

1. Implement the **'SimpleVacuumEnvironment'** class that represents the vacuum cleaner world and includes methods to:
   - Check if a room is dirty.
   - Clean a room.
   - Move the agent to another room.
   - Display the current state of the environment.
2. Implement the **'GoalBasedVacuumAgent'** class that represents the vacuum cleaner agent using a Goal-Based Agent approach. The agent should have methods to:
   - Set goals with actions and priorities.
   - Prioritize goals based on their priorities.
   - Perceive the current room and dirt status.
   - Decide actions based on the highest-priority goal.
   - Perform the decided action.
3. Create an instance for each class and simulate the agent's actions for a few steps, displaying the environment's state after each action.

Problem Statement

Simulate a simple two-room vacuum cleaner world where a vacuum cleaner agent is placed in a house with two rooms (Room A and Room B), and its goal is to clean the dirty rooms.

Environment Description

- The house has two rooms: Room A and Room B.
- Each room can be in one of two states: "Clean" or "Dirty".
- The agent starts in one of the rooms at random.
- The agent can move between rooms and clean the rooms.

Agent Description

- The vacuum cleaner agent uses a Utility-Based Agent approach to make decisions.
- The agent has the following sensors:
  - Current room location (Room A or Room B).
  - Dirt sensor (detects if the current room is dirty).
- The agent maintains utility values for each room, representing the desirability of its state.

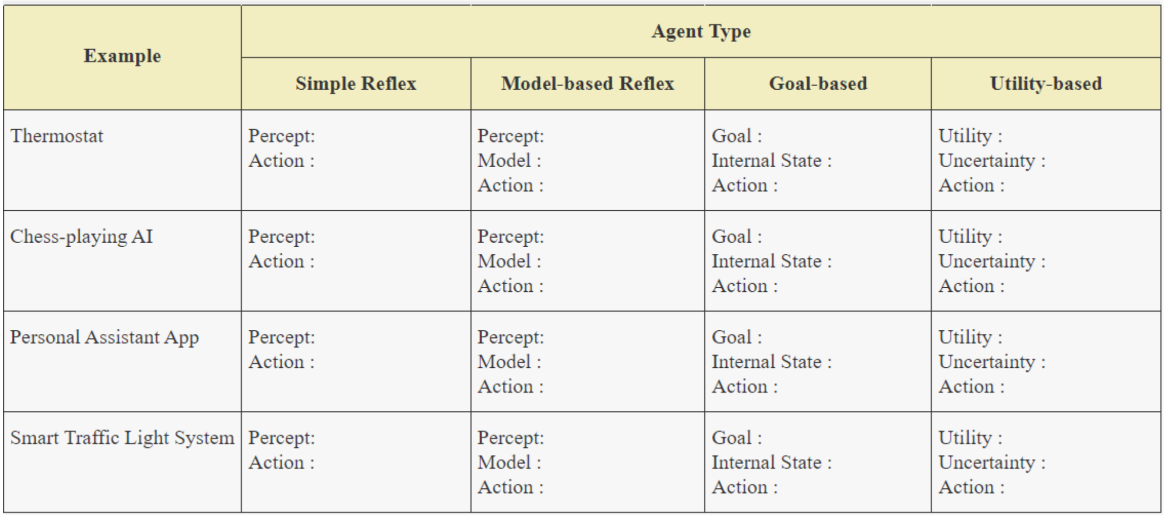
Agent's Decision-Making

- The agent's decision-making is based on utility values assigned to different states:
  - The agent calculates utility values for each room based on its cleanliness.
  - The agent prioritizes actions based on the calculated utilities and selects the action with the highest utility.

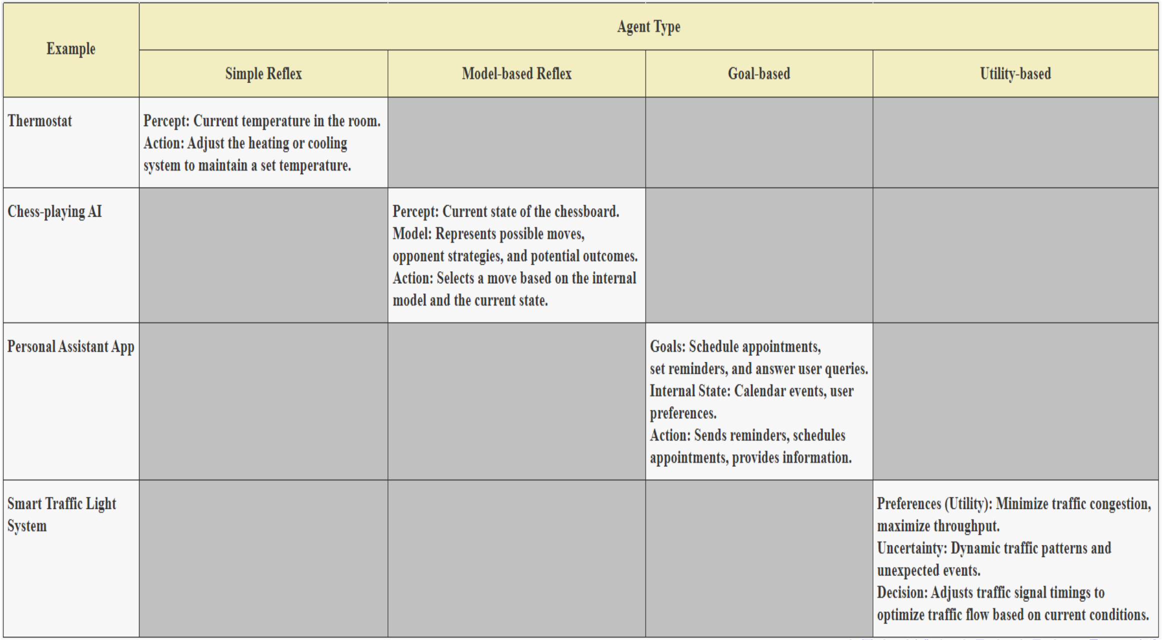
Tasks

1. Implement the **'SimpleVacuumEnvironment'** class that represents the vacuum cleaner world and includes methods to:
   - Check if a room is dirty.
   - Clean a room.
   - Move the agent to another room.
   - Display the current state of the environment.
2. Implement the **'UtilityBasedVacuumAgent'** class that represents the vacuum cleaner agent using a Utility-Based Agent approach. The agent should have methods to:
   - Calculate utility values for each room based on cleanliness.
   - Decide actions based on utility values.
   - Perform the decided action.
3. Create an instance for each class and simulate the agent's actions for a few steps, displaying the environment's state after each action.

Let's fill this table shown together to check how much we understood about Agents.



Example	Agent Type			
	Simple Reflex	Model-based Reflex	Goal-based	Utility-based
Thermostat	Percept: Action :	Percept: Model : Action :	Goal : Internal State : Action :	Utility : Uncertainty : Action :
Chess-playing AI	Percept: Action :	Percept: Model : Action :	Goal : Internal State : Action :	Utility : Uncertainty : Action :
Personal Assistant App	Percept: Action :	Percept: Model : Action :	Goal : Internal State : Action :	Utility : Uncertainty : Action :
Smart Traffic Light System	Percept: Action :	Percept: Model : Action :	Goal : Internal State : Action :	Utility : Uncertainty : Action :



Example	Agent Type			
	Simple Reflex	Model-based Reflex	Goal-based	Utility-based
Thermostat	Percept: Current temperature in the room. Action: Adjust the heating or cooling system to maintain a set temperature.			
Chess-playing AI		Percept: Current state of the chessboard. Model: Represents possible moves, opponent strategies, and potential outcomes. Action: Selects a move based on the internal model and the current state.		
Personal Assistant App			Goals: Schedule appointments, set reminders, and answer user queries. Internal State: Calendar events, user preferences. Action: Sends reminders, schedules appointments, provides information.	
Smart Traffic Light System				Preferences (Utility): Minimize traffic congestion, maximize throughput. Uncertainty: Dynamic traffic patterns and unexpected events. Decision: Adjusts traffic signal timings to optimize traffic flow based on current conditions.

Thank You.