AI Week 2 Notes

Here is the formatted version of your chapter:

Chapter 2: Agents and Environments

Dr. Ammar Masood
Department of Cyber Security, Air University Islamabad

Contents

Agents and Environments

What is an Intelligent Agent?

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

- **Human Agent**: Sensors (eyes, ears, other organs), Actuators (hands, legs, vocal tract).

- **Robotic Agent**: Sensors (cameras, infrared range finders), Actuators (various motors).
- **Software Agent**: Sensors (keystrokes, file contents, network packets), Actuators (screen display, file writing, network communication).

Agent-Environment Cycle

1. **Perception**: The agent receives input (**percepts**) from the environment.
2. **Action**: The agent performs actions based on its current state and knowledge.

Example:

A **robot vacuum cleaner** that senses dirt (**percept**) and moves to clean the floor (**action**).

Vacuum Cleaner Example

This environment consists of two locations: **Square A and Square B**.

- The **vacuum agent** perceives its location and checks if dirt is present.
- It can **move left, move right, suck dirt, or do nothing**.
- A simple agent function:
  - If the current square is **dirty**, then **suck**.
  - Otherwise, **move to the other square**.

Sequence of Actions

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
[B, Clean]	Left
[B, Dirty]	Suck

Structure of Intelligent Agents

Key Components

- **Sensors**: Devices that perceive the environment.
- **Actuators**: Mechanisms that allow the agent to act (motors, speakers, etc.).
- **Agent Function**: Maps percept histories to actions.

Flow:
Environment → [Sensors] → Agent → [Actuators] → Environment

Agent Program

AI designs **agent programs** that implement the **agent function** (mapping from percepts to actions).

$$\text{agent} = \text{architecture} + \text{program}$$

Example: Table-Driven Agent

```plaintext
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
```

How it Works?

1. The agent **observes** the environment and receives a **percept**.
2. It **stores** this percept in a history list.
3. It **looks up** an action from a predefined **table** based on the stored sequence.
4. It **performs** the selected action.

Example: Autonomous vehicles use **sensors** (cameras, radar, LiDAR) to gather data and make driving decisions via **actuators** (steering, brakes).

How Should Agents Act?

Rational Agent

A rational agent always does **the right thing**.

- The **right action** is the one that maximizes the agent's success.
- A **performance measure** evaluates the agent's success based on the effects of its actions.

Example: Vacuum Cleaner Performance Measures

- **Basic Measure**: Amount of dirt cleaned in an 8-hour shift.
- **Advanced Measure**: Efficiency based on power consumption and noise level.
- **Best Measure**: Cleans **quietly** and **efficiently**.

PEAS Factors

PEAS helps define the agent's **task**:

- **P**erformance Measure
- **E**nvironment
- **A**ctuators
- **S**ensors

Example: Autonomous Taxi

Component	Description
Performance	Safety, speed, comfort, profit maximization

Component	Description
Environment	Roads, traffic, pedestrians, customers
Actuators	Steering, brakes, accelerator, horn, signals
Sensors	Cameras, sonar, GPS, speedometer, engine sensors

Environment Parameters

1. Fully Observable vs. Partially Observable

- **Fully Observable**: The agent has access to the **entire environment state**. *(Example: Chess)*
- **Partially Observable**: The agent has **limited** information. *(Example: Taxi driving)*

2. Deterministic vs. Stochastic

- **Deterministic**: The next state is **fully predictable**. *(Example: Chess)*
- **Stochastic**: The next state is **uncertain**. *(Example: Car driving – unpredictable traffic conditions)*

3. Episodic vs. Sequential

- **Episodic**: Agent actions are **independent** of previous actions. *(Example: Spotting defective parts on an assembly line)*
- **Sequential**: Actions affect **future decisions**. *(Example: Chess, Taxi driving)*

4. Static vs. Dynamic

- **Static**: The environment **does not change** while the agent is deciding. *(Example: Solving a crossword puzzle)*
- **Dynamic**: The environment **changes over time**. *(Example: Driving a car)*

5. Discrete vs. Continuous

- **Discrete**: The environment has a **finite set of states**. *(Example: Chess – fixed moves)*

- **Continuous**: The environment has **infinite possibilities**. *(Example: Car driving – speed, steering angle, etc.)*

6. Single-Agent vs. Multi-Agent

- **Single-Agent**: The agent works alone. *(Example: Solving a puzzle)*
- **Multi-Agent**: The agent interacts with others. *(Example: Chess, Taxi driving with pedestrians and other cars)*

Examples of Different Environments

Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Chess (with clock)	Fully	Deterministic	Sequential	Static	Discrete	Multi
Taxi Driving	Partial	Stochastic	Sequential	Dynamic	Continuous	Single
Robot Part Picking	Fully	Deterministic	Episodic	Static	Discrete	Single

---

This formatted version should be **clearer and easier to study** for your exam. Let me know if you need any modifications! 🚀

Here's your content with improved formatting for better readability and study preparation.

Agent Types

> *Each type builds upon the previous, from basic condition–action rules to agents that learn and adapt.*

---

1. Simple Reflex Agent

Agents that act solely based on the current percept, following predefined **condition–action rules**.

- **Example:**
  - If `car-in-front-is-braking`, then `initiate-braking`.
  - A **basic thermostat** that turns on the heater when the temperature drops below a set point.

How They Work?

- Ignore history and internal state.
- Respond immediately to stimuli.

Algorithm (Simple Reflex Agent)

A vending machine operates as a **simple reflex agent**:

- If a user inserts a coin, then dispense the product.
- If the user selects a product, then dispense the change.
- The vending machine responds directly to the current state without considering past experiences or future predictions.

Strengths and Weaknesses

✅ **Strengths:**

- Simple and easy to implement.
- Fast and efficient.
- Suitable for well-defined environments.

❌ **Weaknesses:**

- Limited adaptability.
- Cannot learn from past experiences.
- Requires a **fully observable** environment.

---

2. Model-Based Reflex Agent

Agents that maintain an **internal model (state)** of the world, enabling them to handle **partially observable** environments.

- **Example:**
  - A **robot vacuum** that maps out a room's layout to avoid obstacles and cover the floor efficiently.

How They Work?

- Use a **model** to update their **internal state** based on past percepts.
- Make decisions using both **current percepts** and **stored state**.
- Uses the function **UPDATE-STATE** to maintain an internal model.

Algorithm (Model-Based Reflex Agent)

```plaintext
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
```

Example: Self-Driving Car

A self-driving car has an internal model of the environment, including **road conditions, traffic patterns, and nearby obstacles**.

- If the car detects a **pedestrian crossing**, then **slow down and prepare to stop**.
- If the car detects a **traffic jam ahead**, then **change lanes to avoid congestion**.

Benefits and Drawbacks

✅ **Benefits:**

- Handles **partially observable** environments.
- More **flexible** than simple reflex agents.
- Can use an **internal model** to make **predictions**.

❌ **Drawbacks:**

- Increased complexity.
- Performance relies on **accuracy of the internal model**.
- Limited learning capability.

---

3. Goal-Based Agents

Agents that make decisions by considering the **future consequences** of their actions and strive to achieve **defined goals**.

- **Example:**
  - A **navigation system** that plans a route from **Point A** to **Point B** while avoiding traffic.

How They Work?

- Have a **goal** (or set of goals) that **guides decision-making**.
- Use **search** or **planning techniques** to choose actions that lead to **goal satisfaction**.

Algorithm (Goal-Based Agent)

```plaintext
function GOAL-BASED-AGENT(percept) returns an action
persistent:
  - state, the agent's current conception of the world state
  - goal, the desired goal state
  - model, a description of how the next state depends on current state and action
  - actions, a set of possible actions
  - action, the most recent action, initially none

state ← UPDATE-STATE(state, action, percept, model)
if GOAL-REACHED(state, goal) then
```

```
    return STOP-ACTION()

plan ← SEARCH(state, goal, actions, model)  // Find a sequence of actions to reach
goal
action ← SELECT-ACTION(plan)
return action
```

Example: Robot Navigating a Maze

1. **Goal Definition:** Reach **Point B**.
2. **Planning:** Uses a **planning algorithm** to determine the best path.
3. **Action Selection:** Selects the best action based on planning.
4. **Action Execution:** Executes the selected action.
5. **Percept Feedback:** Adjusts if an obstacle is encountered.
6. **Goal Achievement:** Continues planning until the goal is reached.

Advantages and Weaknesses

✅ **Advantages:**

- Can **adapt** behavior based on the situation.
- Functions in environments with **multiple possible outcomes**.
- Strong **reasoning capability**.

❌ **Weaknesses:**

- **Computationally expensive** planning.
- Defining **clear goals** is crucial for success.
- **Incomplete information** can lead to **flawed planning**.

---

4. Utility-Based Agents

Agents that **not only pursue goals** but also **evaluate how "good" a state is** by assigning a **utility value**.

- **Example:**

- An **investment advisory system** that recommends portfolios based on **risk (utility) and return**.

How They Work?

- Balance **multiple goals** or preferences.
- Choose actions that **maximize overall utility**.
- Use a **utility function** that acts as an **internal performance measure**.

Algorithm (Utility-Based Agent)

```plaintext
function UTILITY-BASED-AGENT(percept) returns an action
persistent:
  - state, the agent's current conception of the world state
  - utility, a function that maps states to a measure of desirability
  - model, a description of how the next state depends on current state and action
  - actions, a set of possible actions
  - action, the most recent action, initially none

state ← UPDATE-STATE(state, action, percept, model)
if GOAL-REACHED(state) then
    return STOP-ACTION()

best_action ← argmax a ∈ actions UTILITY(RESULT(state, a))
return best_action
```

Example: Financial Advisor App

1. Defines its **goal** as maximizing returns.
2. Uses a **utility function** to evaluate investment options.
3. Selects the **best** investment option.
4. Executes the decision.
5. Receives feedback and adjusts future decisions.

Benefits and Limitations

✅ **Benefits:**

- **Flexible and adaptive**.

- Considers **risk, time, and effort**.

❌ **Limitations:**

- **Complex** utility function design.
- **Computationally expensive** evaluations.
- Uncertainty in **outcomes**.

---

5. Learning Agents

Agents that improve their performance **over time** by learning from **experience**.

- **Example:**
  - An **AI-based game-playing agent** that learns strategies by playing multiple rounds.

How They Work?

- Include **learning and adaptation**.
- Use techniques from **reinforcement learning, supervised learning, or unsupervised learning**.

Components of a Learning Agent

1. **Critic:** Evaluates agent's performance and provides feedback.
2. **Learning Element:** Uses feedback to improve decision-making.
3. **Problem Generator:** Introduces new situations for exploration.
4. **Performance Element:** Makes decisions based on learned knowledge.

Benefits and Drawbacks

✅ **Benefits:**

- Can **adjust to new environments**.
- Handle **complex tasks**.
- **Real-world applicability**.

❌ **Drawbacks:**

- Require **large amounts of data**.
- Need to balance **exploration vs. exploitation**.
- Hard to interpret decisions.

Conclusive Summary of Agent Types

Agent Type	Key Characteristics	Example
Simple Reflex	Reacts to **current percepts**, uses fixed rules	**Basic thermostat**
Model-Based Reflex	Maintains **internal state**, updates based on past percepts	**Robot vacuum**
Goal-Based	Considers **future consequences**, uses search/planning	**Navigation system**
Utility-Based	Maximizes **overall utility**, evaluates multiple outcomes	**Investment advisor**
Learning Agent	**Adapts and improves** over time	**AI game-playing bot**