2. Intelligent Agents

Chapter_2

Introduction to Agents

- 🔷 **What is an Agent?**
- An **agent** is any entity that can **perceive** its environment through sensors and **act** upon that environment through actuators.
- **Agent = Perception + Action**
- 🔷 **Intelligent Agent:**
- An **intelligent agent** is an autonomous system that **perceives** the environment, **reasons** about what actions to take, and then **acts** to achieve specific goals efficiently.

- An **agent** is an entity that **perceives** its environment through **sensors** and **acts** upon that environment using **actuators** to achieve certain goals.
- **Formal Definition**:

An agent is a system that maps **percepts** (input from the environment) to **actions** (output to the environment).

Agent Function and Agent Program

- **Agent Function (f):**

  Mathematical function that maps a **percept sequence** to an action.

  $$f : P^* \to A$$

  Where:

  - P^*: Set of all percept sequences
  - A: Set of actions

- **Agent Program:**

  The **implementation** (e.g., in code) of the agent function running on physical architecture.

Structure of an Agent

- An agent has four core components:

Component	Description
Sensors	Perceive the environment (e.g., camera, microphone)
Actuators	Perform actions in the environment (e.g., motors, speakers)
Percept	A piece of information sensed from the environment
Agent Architecture	Includes hardware and software that runs the agent program

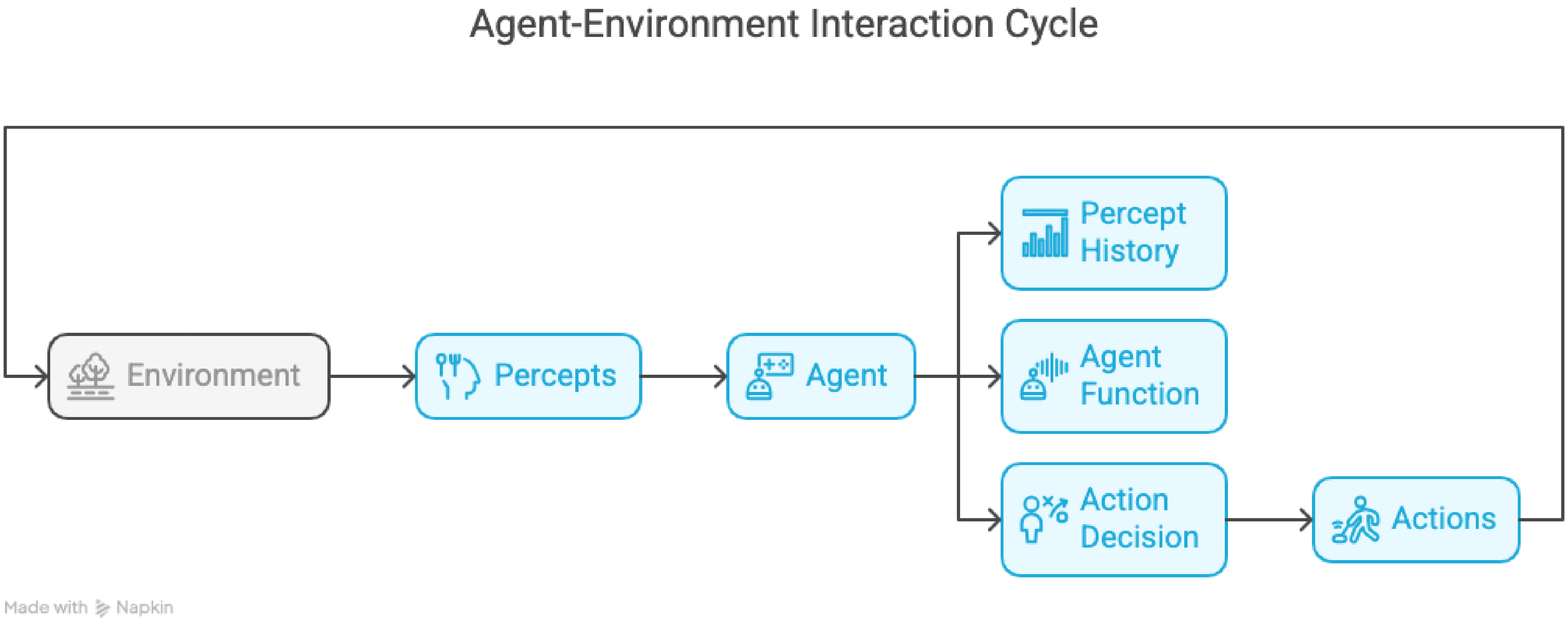
Agent Properties (Desirable Characteristics)

Property	Description
Autonomy	Acts without human intervention
Reactivity	Responds in real-time to changes
Pro-activeness	Takes initiative to achieve goals
Social Ability	Can interact with humans or other agents
Adaptability	Learns and improves over time
Mobility	(In mobile agents) can move through environments/networks

Agent vs. Intelligent Agent

Feature	Basic Agent	Intelligent Agent
Decision Making	Simple rules	Reasoning + learning
Environment Handling	Static, known	Dynamic, uncertain
Goal Orientation	Often reactive	Goal-directed
Adaptation	Limited or none	Can adapt using AI techniques

Agent–Environment Interaction Cycle



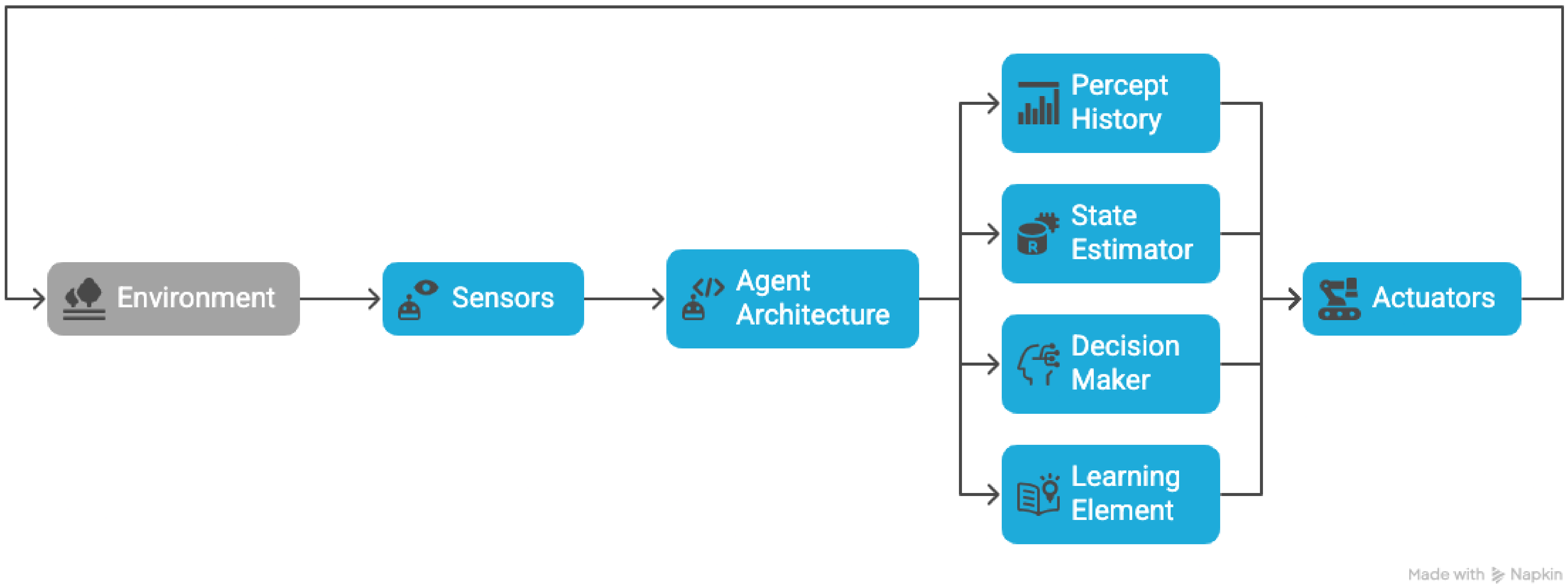
Examples of Agents in AI

Application	Sensors	Actuators	Description
Self-driving car	Cameras, GPS, LIDAR	Steering, brakes	Navigates roads, avoids obstacles
Chatbot (e.g., ChatGPT)	Text input	Text output	Answers user queries naturally
Autonomous drone	Accelerometer, GPS	Motors, camera	Delivers packages or monitors areas
Smart thermostat	Temperature sensor	Heating/cooling	Adjusts room temperature automatically
Industrial robot	Visual sensors	Robotic arms	Assembles parts on a production line

Structure of an Intelligent Agent

- An **Intelligent Agent** is more than just a simple rule-based system. It has internal components and logic that allow it to **reason**, **learn**, and **act** in a dynamic environment.

Agent Interaction with Environment



Key Components

Component	Description
Sensors	Devices or modules used to gather data from the environment (e.g., camera, LIDAR, keyboard).
Actuators	Tools through which the agent acts upon the environment (e.g., motors, display screen, speakers).
Percept History	A log or memory of what the agent has perceived so far (helps with reasoning over time).
State Estimator	Determines the current "state" of the world from percept history and internal model.
Decision Maker	Determines what action to take based on current state, goals, and knowledge (can involve search, planning, rules, etc.).
Learning Element	Optional but powerful: allows the agent to improve over time using machine learning.

Agent Program vs. Agent Function

- **Agent Function**: Abstract mapping from percepts to actions.
- **Agent Program**: The implementation (code/logic) that realizes this function.

Properties of Intelligent Agents

- **1. Autonomy**
- The agent operates **independently** without continuous human intervention.
- It makes decisions on its own and can adjust its behavior based on its experiences.
- **Example**: A Mars Rover deciding which rock to analyze next.
- **2. Reactivity**
- It perceives its environment and **responds** to changes **in real-time**.
- **Example**: A security bot that changes patrol route when it detects movement.

- **3. Proactiveness (Goal-Directed Behavior)**
- Not only reactive but also **takes the initiative** to achieve its objectives.
- **Example**: A personal assistant AI proactively scheduling meetings based on your preferences.
- **4. Social Ability**
- Can interact with **humans** or **other agents** using communication protocols or natural language.
- **Example**: AI bots in a multi-agent negotiation system or customer service chatbots.

- **5. Adaptability (Learning Ability)**
- Uses **learning algorithms** to **improve** behavior or decision-making over time.
- **Example**: A spam filter that learns new spam trends by analyzing incoming emails.
- **6. Rationality**
- Chooses actions that are expected to **maximize performance** given what it knows.
- **Example**: A self-driving car weighing all options to minimize accident risk and time delay.
- **7. Mobility *(optional but important in physical agents)***
- Ability to **move within environments** or even across networks in software agents.
- **Example**: A mobile drone agent scanning disaster-hit zones.

Example: Intelligent Assistant (e.g., Siri, Alexa)

Property	Implementation Example
Autonomy	Responds without direct programming for every task
Reactivity	Answers based on voice commands or alerts
Proactiveness	Suggests reminders or news based on habits
Social Ability	Engages in natural language dialogue
Adaptability	Learns user preferences over time
Rationality	Picks best response based on user intent
Mobility	(Not applicable – software-only agent)

Relationship Between Agents and Environments

- In Artificial Intelligence, an **agent** operates within an **environment**. The agent's **goal** is to perceive the environment and take actions that **maximize performance** or **achieve goals**.
- **Fundamental Concept**:
- **Agents sense** the environment using **sensors** and **act** upon it using **actuators**. The **interaction loop** is continuous and dynamic.

Components of Agent–Environment Interaction

Component	Description
Agent	The decision-making entity (software, robot, etc.)
Environment	The external world or context the agent operates in
Sensors	Tools to perceive the environment (e.g., camera, microphone)
Actuators	Mechanisms to act on the environment (e.g., motors, speaker)
Percepts	Input data from the environment
Actions	Output actions taken by the agent

Key Considerations in Agent–Environment Relationship

- **1. Perceptual Input**
- The agent's **perception** is a limited view of the environment.
- **Quality of sensors** affects decision-making.
- **Example**: A robot's camera might misinterpret shadows as obstacles.
- **2. Actions and Effects**
- Every **action** taken by the agent **modifies** the environment in some way.
- These changes are **observed again** in the next cycle.
- **Example**: A drone flying upward gets closer to the obstacle it must avoid.

Key Considerations in Agent–Environment Relationship

- **3. Environment Feedback**
- The **state of the environment** may change due to:
  - Agent’s actions
  - Other agents
  - Natural or external factors
- **Example**: In a self-driving car scenario, another vehicle cutting in is an external environmental change.

Types of Environments

- Environments can be classified based on **how they affect agent design**:

Environment Type	Description	Implication for Agent
Fully Observable	Agent has access to **complete** state of environment	Easier to design; no internal state needed
Partially Observable	Agent sees only **part** of the environment	Needs internal model or memory
Deterministic	Next state is completely determined by current state and action	Predictable planning is possible

Environment Type	Description	Implication for Agent
Episodic	Agent’s experience is **divided into episodes**; decisions are independent	No long-term strategy needed
Sequential	Current decision affects **future decisions**	Requires planning
Static	Environment **doesn’t change** during agent’s action	Easier reasoning
Dynamic	Environment **changes over time**, even without the agent	Needs real-time decision-making
Discrete	Finite number of percepts/actions	Simpler logic
Continuous	Infinite possibilities in time/state/action	Needs approximation or continuous models
Single-Agent	Only one agent acting in the environment	Strategy depends only on itself
Multi-Agent	Other agents present (cooperative or competitive)	Game theory, negotiation needed

Examples of Agent–Environment Pairs

Agent	Environment	Sensor Input	Actuator Output
Self-driving car	Roads, traffic, pedestrians	LIDAR, GPS, cameras	Steering, throttle, brakes
Chatbot	Text chat interface	Text input	Text response
Vacuum robot	House floorplan, obstacles	Infrared, bumper sensors	Wheel motors
Trading agent	Stock market	Market data	Buy/Sell signals

Types of Agents in AI

- Agents vary in complexity based on their internal architecture and reasoning capabilities. The four primary types of intelligent agents are:

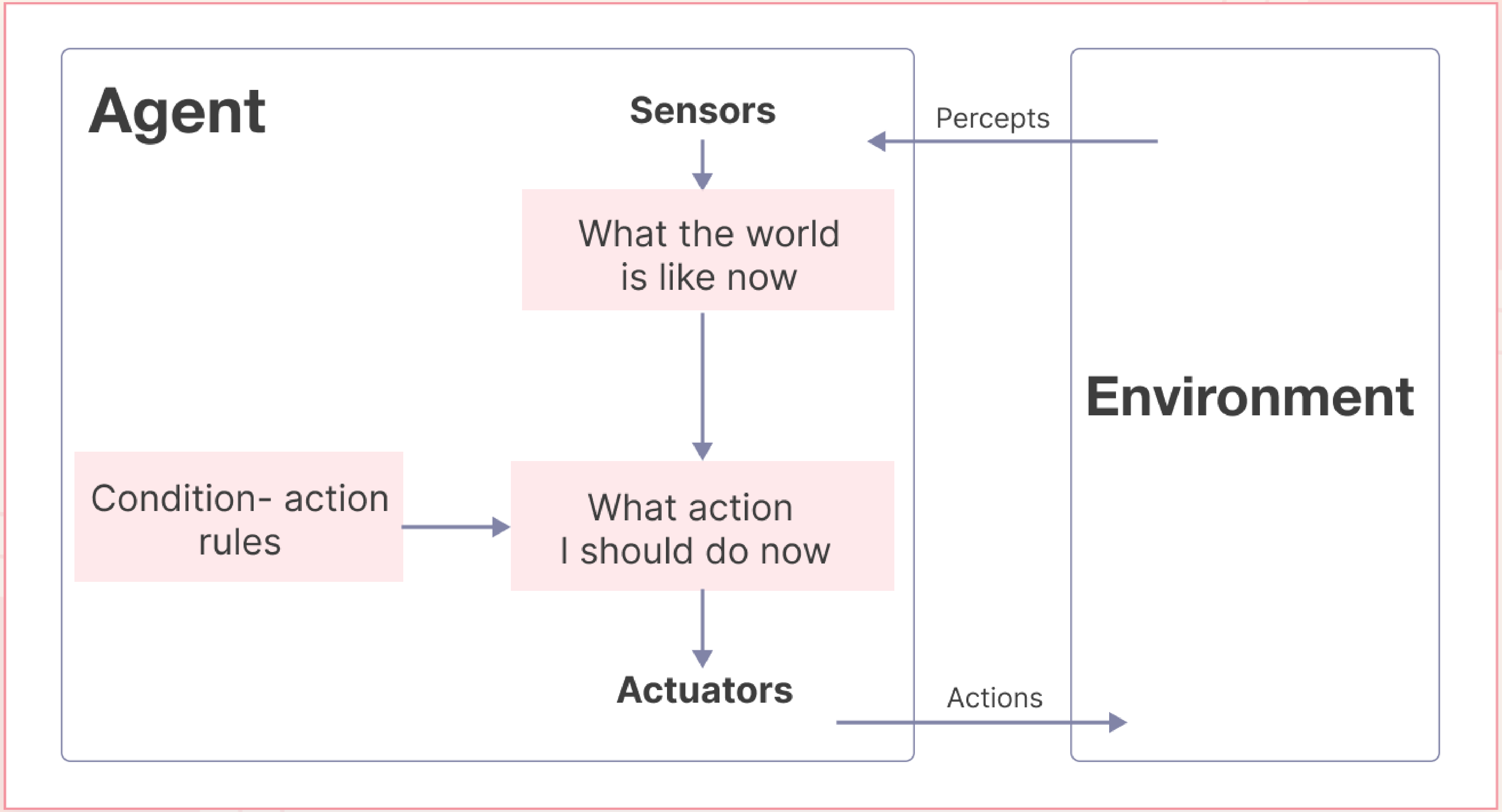
1. **Simple Reflex Agents**
2. **Model-Based Reflex Agents**
3. **Goal-Based Agents**
4. **Utility-Based Agents**

Simple Reflex Agent

- A **Simple Reflex Agent** selects actions based solely on the **current percept** and **predefined condition–action rules**.
  It does **not** retain memory or internal state and **does not reason** beyond the current input.
- In essence:

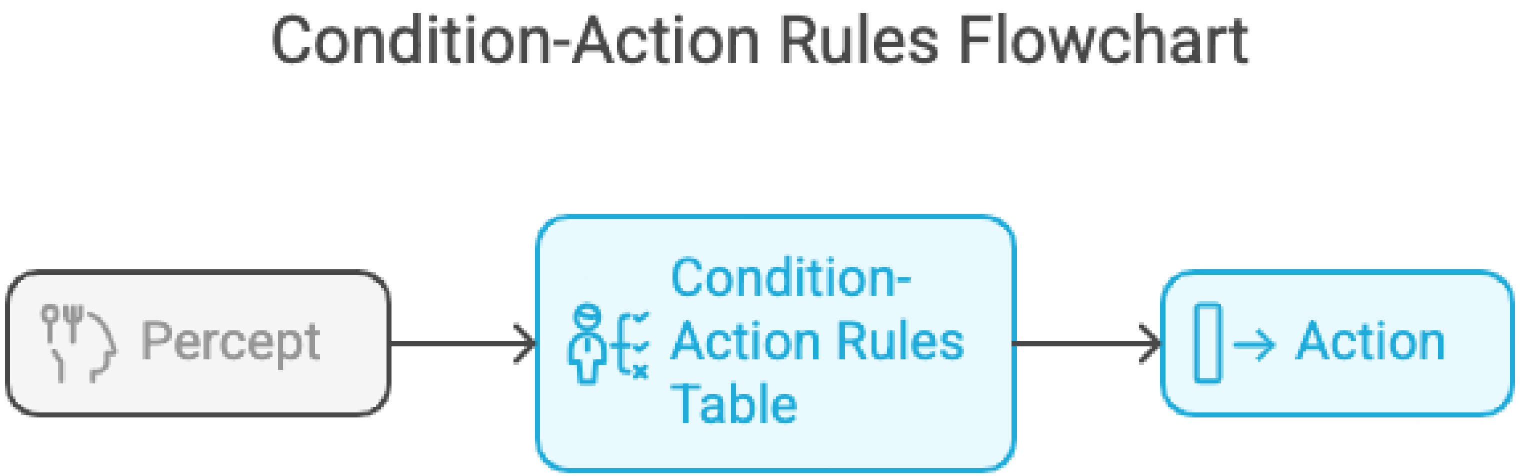
"If this percept is seen, then perform that action."

Simple Reflex Agent



SIMFORM

How It Works



The agent consults a **set of rules** to determine what action to take for each percept.

Each rule looks like:

if percept == X, then do action Y

Key Characteristics

Feature	Description
Stateless	No memory of past percepts
Reactive	Responds instantly to input
Deterministic	Always chooses the same action for the same input
Efficient	Fast and low-resource
No learning	Cannot adapt or improve over time

Example 1: Thermostat Agent

Percept	Action
Temp < 20°C	Turn on heater
Temp ≥ 20°C	Turn off heater

Condition-Action Rule:

if temperature < threshold → turn heater on

This thermostat does not "remember" previous temperatures or anticipate future ones—it just reacts.

Example 2: Vacuum Cleaner Agent

Percept	Action
"Dirt detected"	Suck dirt
"Obstacle ahead"	Turn right
"Clear path"	Move forward

Simple behavior, based on **immediate sensory input** only.

Advantages

Advantage	Description
Simplicity	Easy to design and implement
Speed	Fast response (no computation delay)
Low resource	Minimal hardware/software needs

Works well in fully observable, static environments

Limitations

Limitation	Impact
No memory	Can’t reason or plan
Not adaptive	Can’t improve with experience
Fails in complex settings	Not suitable for dynamic or partially observable environments
Not goal-directed	Has no understanding of purpose or long-term objectives

Behavioral Model

- **Behavior = f(percept)**
- It does not consider **time**, **history**, or **future consequences**.

Use Case Suitability

Environment Type	Suitability
Fully observable	✅ Good
Partially observable	❌ Poor
Static	✅ Good
Dynamic	❌ Poor
Episodic	✅ Good
Sequential	❌ Poor

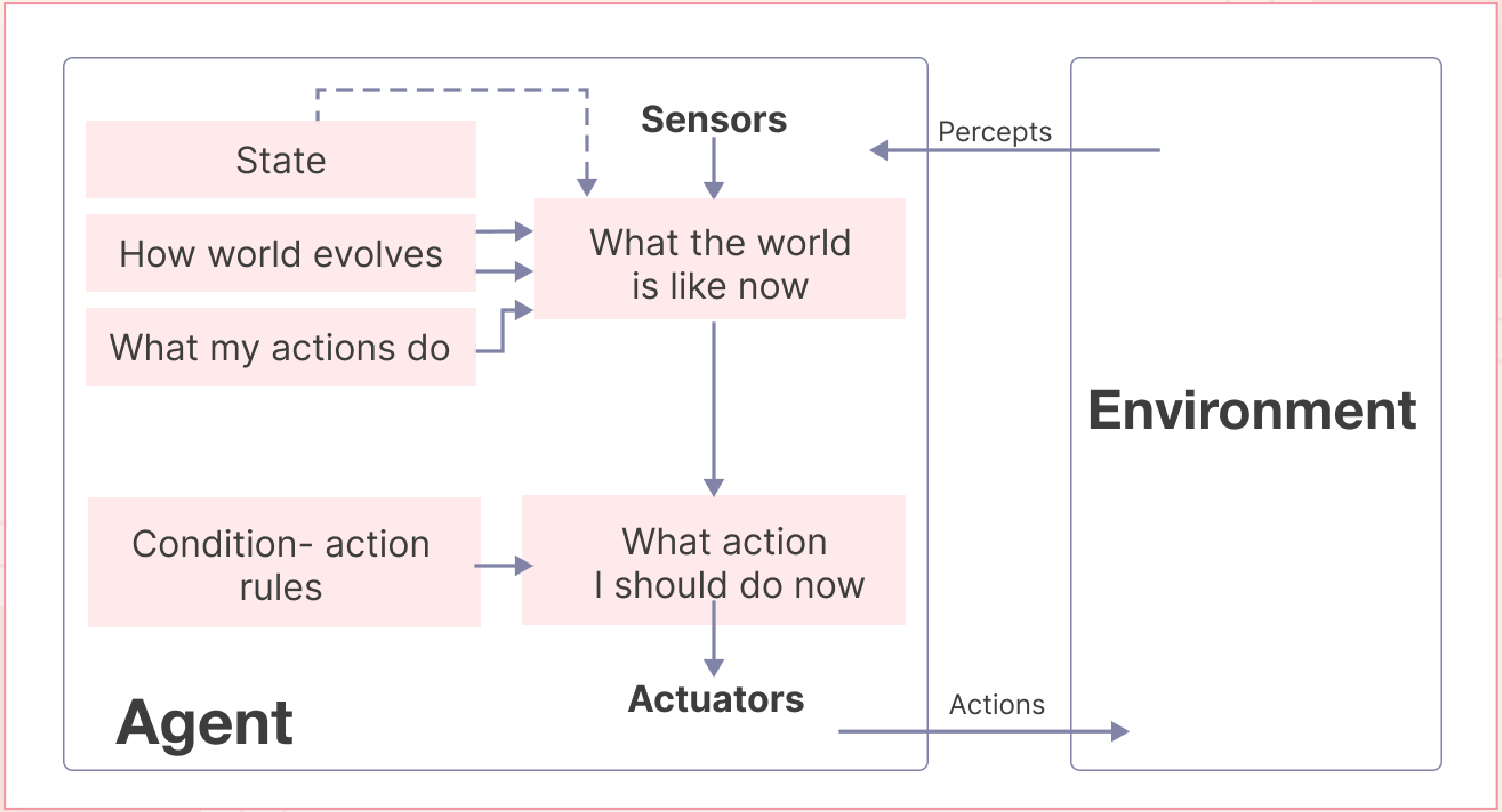
When to Use Simple Reflex Agents

- When tasks are **straightforward** and **environments are predictable**.
- Example domains:
  - **Embedded control systems** (e.g., smart switches, alarms)
  - **Preprogrammed robots** (e.g., toy robots, line-followers)
  - **Simple NPCs** in games (e.g., patrolling guards)

Model-Based Reflex Agent

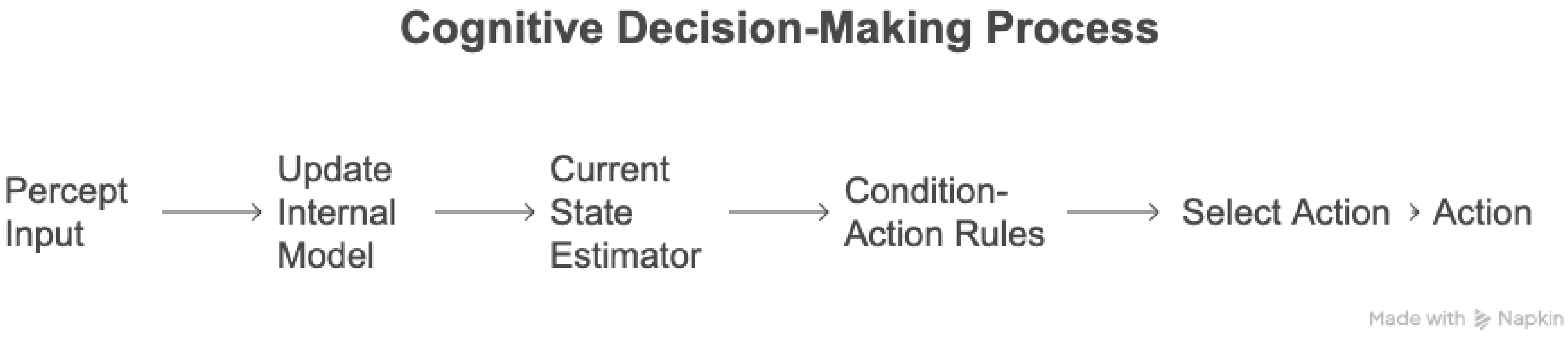
- A **Model-Based Reflex Agent** is an intelligent agent that uses an **internal model of the environment** to handle **partially observable** situations.
- **Key Idea**:
- It maintains an **internal state (memory)** to track aspects of the world that cannot be directly observed at every moment.

Model-Based Reflex Agent



SIMFORM

Architecture of Model-Based Reflex Agent



How It Works

- The **model** represents the agent’s understanding of:
  - **How the environment evolves** (state transition model).
  - **How percepts relate to environment states** (observation model).
- Example: A vacuum robot knows that if it moves forward, it changes its position, even if no new percept comes in.

Differences from Simple Reflex Agent

Feature	Simple Reflex Agent	Model-Based Reflex Agent
Memory of past percepts	❌ No memory (stateless)	✅ Maintains internal state (model)
Environment type handled	Fully observable, static only	Partially observable, dynamic
Complexity	Very simple decision-making	Requires model and state estimation
Adaptability	Cannot adapt to unseen situations	Can infer unobserved states based on experience

Example Scenario: Vacuum Cleaning Robot

- **Problem:**
- The robot cannot always “see” the entire floor.
- Sensors may only detect dirt **directly below** it.
- **Solution (Model-Based Agent):**
- Maintains an **internal map** of cleaned/uncleaned areas.
- Tracks **its current position** and **where it has been**.
- Uses this to infer which areas are likely still dirty.
- **Working:**
- Updates internal state as it moves.
- Uses rules like:
  - *"If adjacent tile is dirty → Move there and clean."*
  - *"If all known tiles are clean → Explore new areas."*

Agent Loop (Detailed)

1. Sense → "Perceive dirt at current location."
2. Update → "Mark this tile as clean in internal map."
3. Decide → "Choose next tile to clean based on map."
4. Act → "Move to selected tile and repeat."

Advantages

Advantage	Description
Handles partial observability	Can infer missing information using internal model.
More intelligent behavior	Appears smarter than simple reflex agents.
Efficient operation	Avoids redundant or wasteful actions.
Scalable	Can handle larger, dynamic environments.

Limitations

Limitation	Impact
Increased complexity	Requires state tracking and model updating logic.
Still lacks goals/utility	Cannot plan long-term sequences of actions without goal-based extensions.
Depends on model accuracy	Faulty models can lead to poor decisions.

Real-World Applications

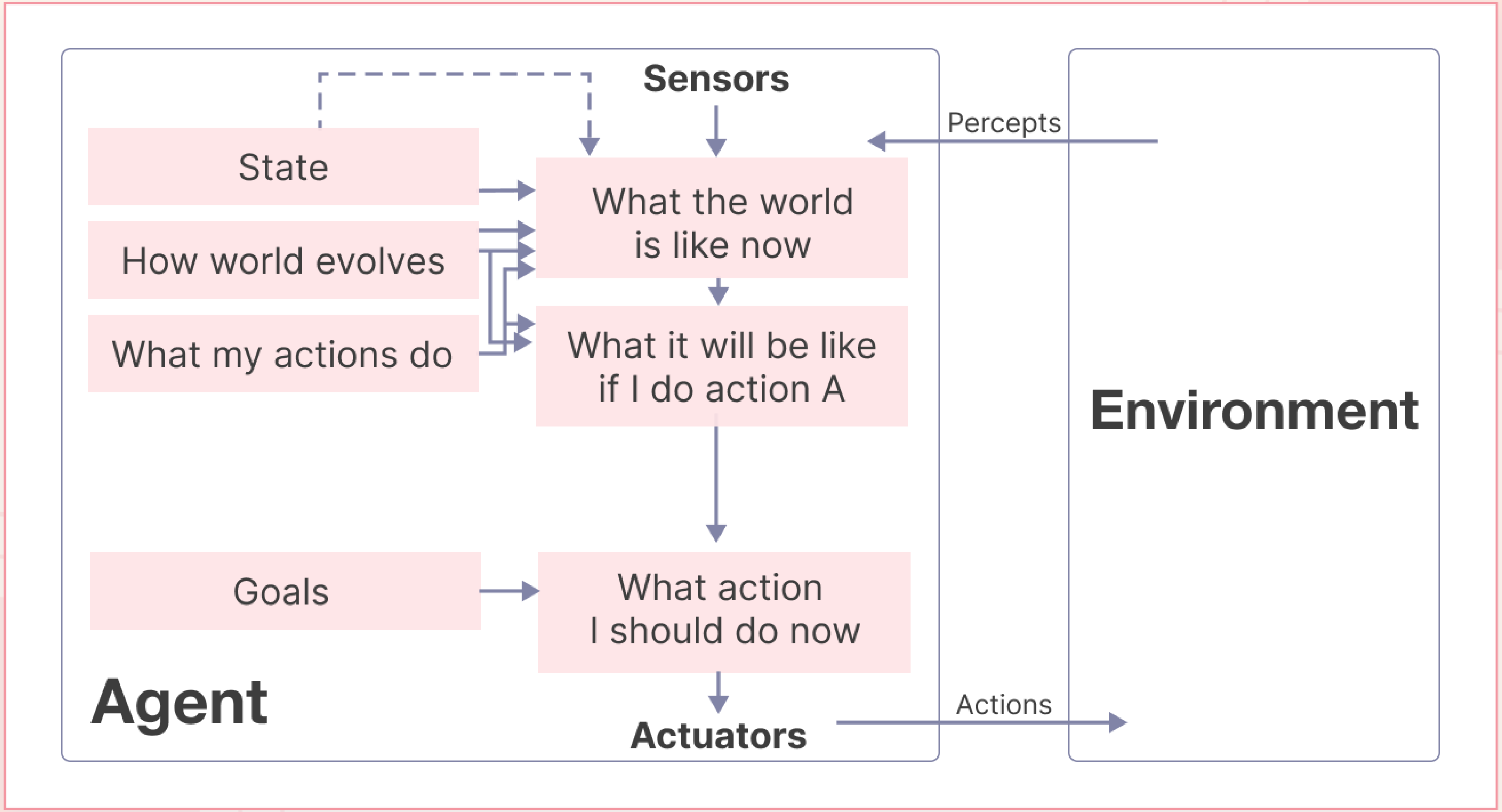
Application	Model-Based Reflex Use
Robotic vacuum cleaners	Remembering cleaned vs. uncleaned areas.
Mobile navigation bots	Tracking position in partially mapped environments.
Simple warehouse robots	Keeping track of picked/delivered items.
Security patrol drones	Maintaining visited/unvisited area memory.

Goal-Based Agent

- A **Goal-Based Agent** is an intelligent agent that **makes decisions** by considering **goals** it wants to achieve. Rather than just reacting to the current situation, it **deliberates**, **searches**, and **plans** actions to achieve a **desired future state**.
- Key idea:

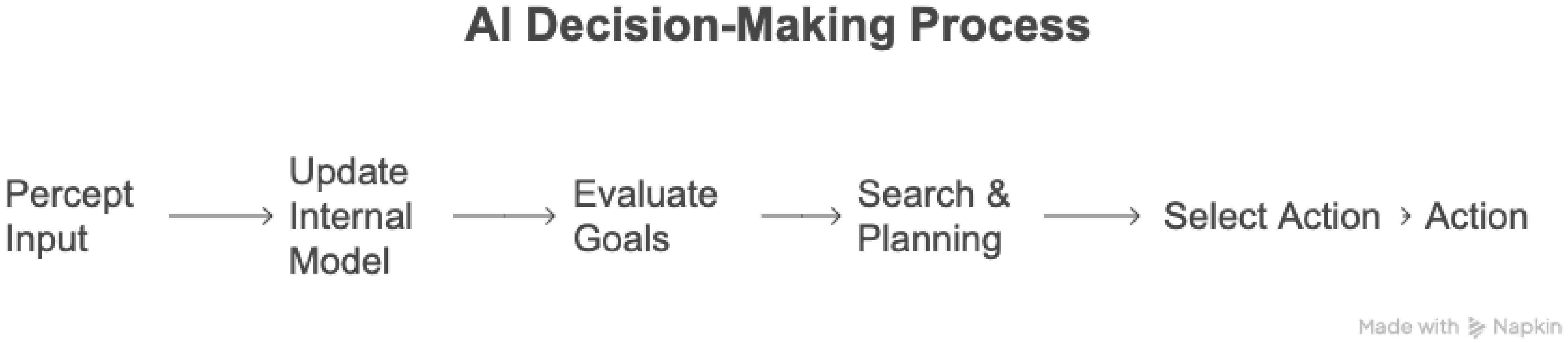
"Think before you act" — the agent evaluates different possibilities and selects actions that lead to its goal.

Goal-based Agent



SIMFORM

Architecture of Goal-Based Agent



Core Components

Component	Role
Percepts	Input from environment
Internal State	Model of the current world
Goals	Representation of desirable states
Search Mechanism	Finds sequences of actions to reach the goal
Planner	Organizes actions in order of execution

How It Works

- **Perceive**: Sense environment data through sensors.
- **Update Internal State**: Maintain knowledge of the world.
- **Define Goal(s)**: Understand desired outcomes (e.g., reach destination, clean room).
- **Search & Plan**:
- Analyze possible actions.
- Predict their consequences.
- Formulate a plan (sequence of actions) to reach the goal.
- **Action Execution**: Carry out the chosen action(s).

What is a Goal?

- A **goal** is a **desired state of the environment**.
- Represented formally in logical expressions, search trees, or rule-based formats.
- **Example**:
  Goal: *“Agent should reach location (x, y)”*.

Example Scenario: Self-Driving Car

- **Goal: Reach Destination Safely and Quickly**
- **Percepts**: Road conditions, GPS location, traffic, pedestrian movement
- **Goal**: Arrive at a destination
- **Search**: Use pathfinding algorithms (e.g., A*, Dijkstra) to plan a route
- **Action**: Drive, stop, turn, accelerate, or reroute if needed

Example Rule

IF goal = "Reach Office"

AND current_location ≠ office

THEN plan_path(current_location → office)

AND execute(path)

Why Goals Matter

- **Goals guide decision-making.**
- Agents can **evaluate** alternative courses of action and **choose the best one**.
- This creates **flexibility** and **intelligence** not possible in reflex agents.

Techniques Used

- **Search Algorithms**: BFS, DFS, A*, Iterative Deepening
- **Planning Algorithms**:
- STRIPS (Stanford Research Institute Problem Solver)
- Hierarchical Task Networks (HTN)
- Partial Order Planning
- **Heuristic Evaluation**: Guides the agent in choosing efficient paths

Real-World Applications

Application	Example Use
Self-driving cars	Goal: Reach destination safely and efficiently.
Autonomous drones	Goal: Complete delivery, avoid obstacles.
Game AI (NPC agents)	Goal: Defeat player, capture objectives.
AI in logistics	Goal: Optimize warehouse order picking routes.
Personal Assistant AI	Goal: Schedule meetings without conflicts.

Advantages

Advantage	Description
Goal-directed behavior	Focused on achieving specific desired outcomes.
Flexible and dynamic	Can adjust to new goals or changing environments.
Supports planning and search	Can select optimal action sequences to achieve goals.
Higher intelligence	Exhibits rational decision-making, beyond reflexes.

Limitations

Limitation	Impact
Computationally expensive	Planning and search can be time-consuming.
Requires accurate world models	Needs reliable information to plan effectively.
Cannot handle trade-offs	Chooses any goal-achieving plan, even if suboptimal (no utility evaluation).
Short-sighted if not optimized	Without utility-based reasoning, may select inefficient paths.

Utility-Based Agent

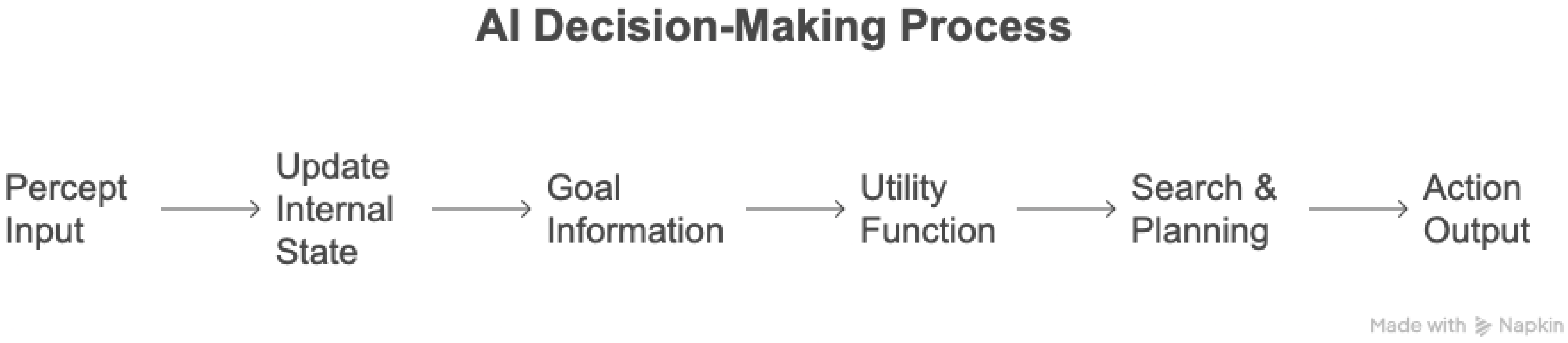
- A **Utility-Based Agent** is an advanced form of intelligent agent that selects actions not just to achieve goals but to **maximize a utility function**, which represents the agent's **degree of satisfaction or performance**.
- **Key Concept**:

Not all goal-achieving states are equally desirable. Utility-based agents choose the **best possible action** based on preferences and trade-offs.

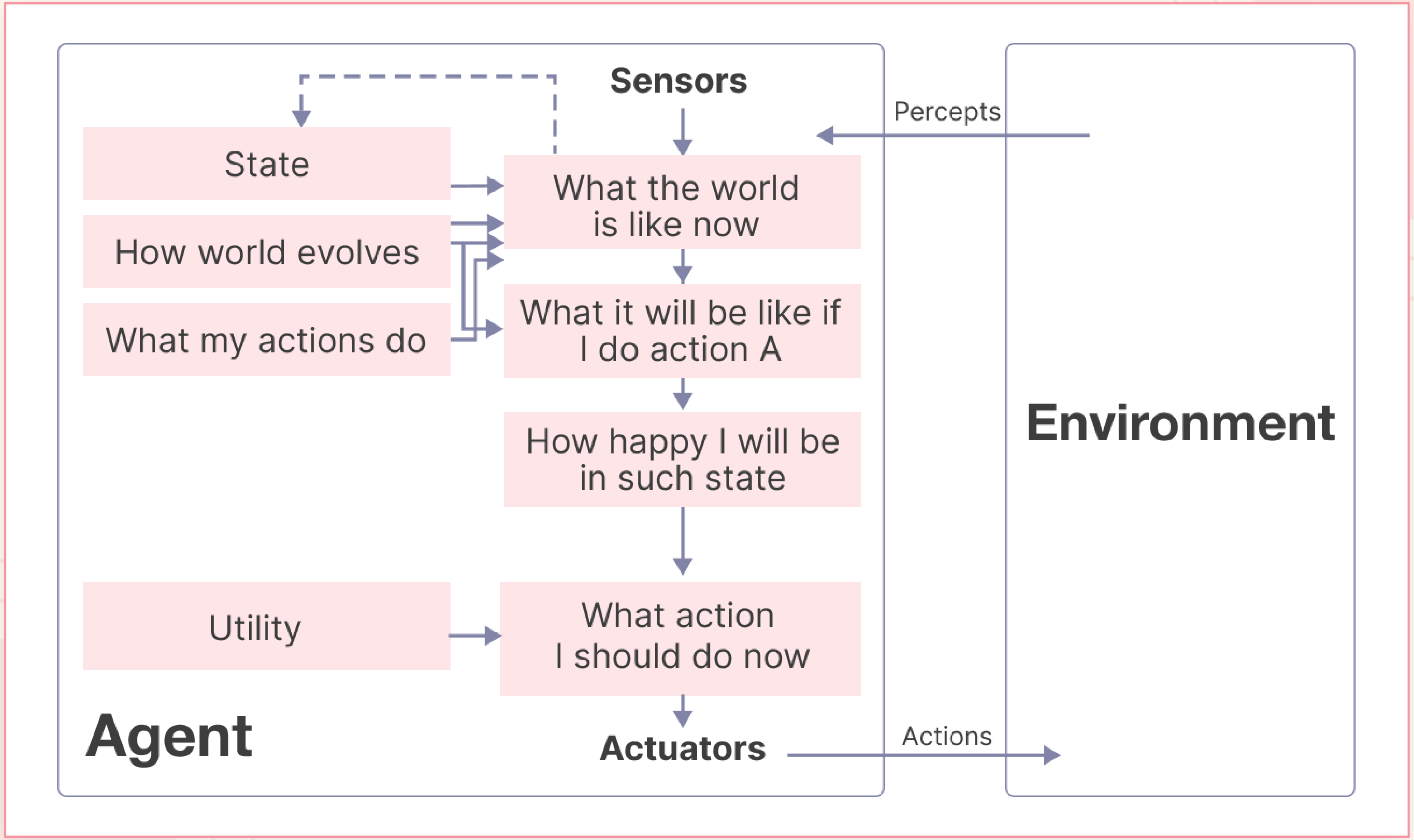
What is Utility?

- **Utility** is a numerical measure of **how much an agent values a particular state**.
- Utility functions quantify **preferences** over possible states.
- Helps in decision-making under **uncertainty** and **multiple conflicting goals**.
- Example: A self-driving car may prefer routes with less traffic even if all options reach the destination (goal).

Utility-Based Agent Architecture



Utility-based Agent



SIMFORM

Working of a Utility-Based Agent

- **Perceive** environment and update internal state.
- **Evaluate goals** and potential outcomes.
- **Compute utility values** for possible actions/states.
- **Search & plan** to find action sequences that maximize utility.
- **Select the action** with the **highest utility**.

Utility Function Example

For a **self-driving car**:

$$Utility(State) = 0.6 \times Safety + 0.3 \times TimeEfficiency + 0.1 \times Comfort$$

- Safety is weighted highest.
- Time efficiency and comfort are secondary but considered.

Advantages

Advantage	Benefit
Optimizes performance	Chooses the best action among alternatives
Handles conflicting goals	Can balance trade-offs (e.g., speed vs. safety)
Decision-making under uncertainty	Uses probabilistic utility estimation
Flexible and adaptable	Adjusts to changing preferences or situations

Limitations

Limitation	Challenge
Complex to design	Requires defining accurate utility functions
Computationally expensive	Needs extensive evaluation of possible states
Model dependency	Requires reliable models for prediction

Real-World Application

Domain	Utility-Based Agent Use
Autonomous Vehicles	Choosing optimal route balancing safety, time, fuel
Recommendation Systems	Suggesting content that maximizes user satisfaction
Financial Trading Bots	Optimizing trade-offs between profit and risk
Game AI	Selecting moves maximizing chance of victory
Healthcare Diagnosis	Recommending treatments balancing effectiveness and side-effects

Example: Investment Decision Agent

- **Goal:** Invest money
- **Utility Function:**

$$Utility = 0.5 \times ExpectedReturn - 0.4 \times Risk + 0.1 \times Liquidity$$

- The agent evaluates options (stocks, bonds, real estate) and selects the one with the highest utility value.

Performance Evaluation of Agents: PEAS Framework

What is PEAS?

- PEAS is a **structured framework** for defining an agent's task environment in AI.
  - **P** – Performance Measure
    E – Environment
    A – Actuators
    S – Sensors
- It's a **blueprint** for designing and evaluating intelligent agents.

1. Performance Measure (P)

- **Defines success** criteria.
- Objectively quantifiable.
- Should avoid rewarding the wrong behavior.
- Drives the agent's actions towards desired outcomes.
- **Good performance measures are aligned with the user's goals**

3. Actuators (A)

- The agent’s **output devices**.
- Responsible for affecting the environment.
- Physical or virtual actions.

4. Sensors (S)

- The agent’s **input devices**.
- Used to perceive the environment.
- Provide data for decision-making.

2. Environment (E)

- **Where the agent operates**.
- Includes physical surroundings, other agents, dynamic factors.
- Can be simple (board game) or complex (real world).

Example 1: Self-Driving Car Agent

PEAS Component	Description
Performance Measure	Safety (avoid collisions), travel time, fuel efficiency, comfort, obey traffic laws
Environment	Roads, traffic lights, other vehicles, pedestrians, weather conditions
Actuators	Steering wheel, accelerator, brakes, indicators, horn
Sensors	Cameras (vision), LIDAR, GPS, Radar, ultrasonic sensors

Example 2: Vacuum Cleaning Robot

PEAS Component	Description
Performance Measure	Area cleaned, energy used, time taken, obstacle avoidance
Environment	Rooms, furniture, dirt patches, stairs
Actuators	Wheels (movement), suction motor, rotating brushes
Sensors	Dirt sensors, bump sensors, cliff sensors, gyroscope

Example 3: Intelligent Personal Assistant (e.g., Siri, Alexa)

PEAS Component	Description
Performance Measure	Response accuracy, speed, user satisfaction, task completion
Environment	User's speech, queries, internet, connected devices
Actuators	Voice output, device control APIs (lights, calendar apps)
Sensors	Microphone, touch input, network access

Example 4: Online Recommendation System (e.g., Netflix, Amazon)

PEAS Component	Description
Performance Measure	Click-through rate, user watch time, purchase rate, satisfaction
Environment	User profiles, preferences, content library, social influence
Actuators	Content recommendations (UI elements), notifications
Sensors	User browsing data, clicks, watch history, ratings

Why PEAS is Essential for Performance Evaluation

- Clarifies **what exactly the agent must optimize**.
- Helps define **relevant environment factors**.
- Determines **what hardware/software is needed**.
- Ensures the agent's design is **aligned with success criteria**.

When designing agents, PEAS ensures:

- **No ambiguity** in task definition.
- The agent’s **sensors/actuators are sufficient**.
- **Evaluation metrics** are aligned with real-world expectations.
- Developers can **objectively assess** agent performance.

"PEAS is like a problem specification sheet for AI agents—ensuring we design and evaluate them systematically."