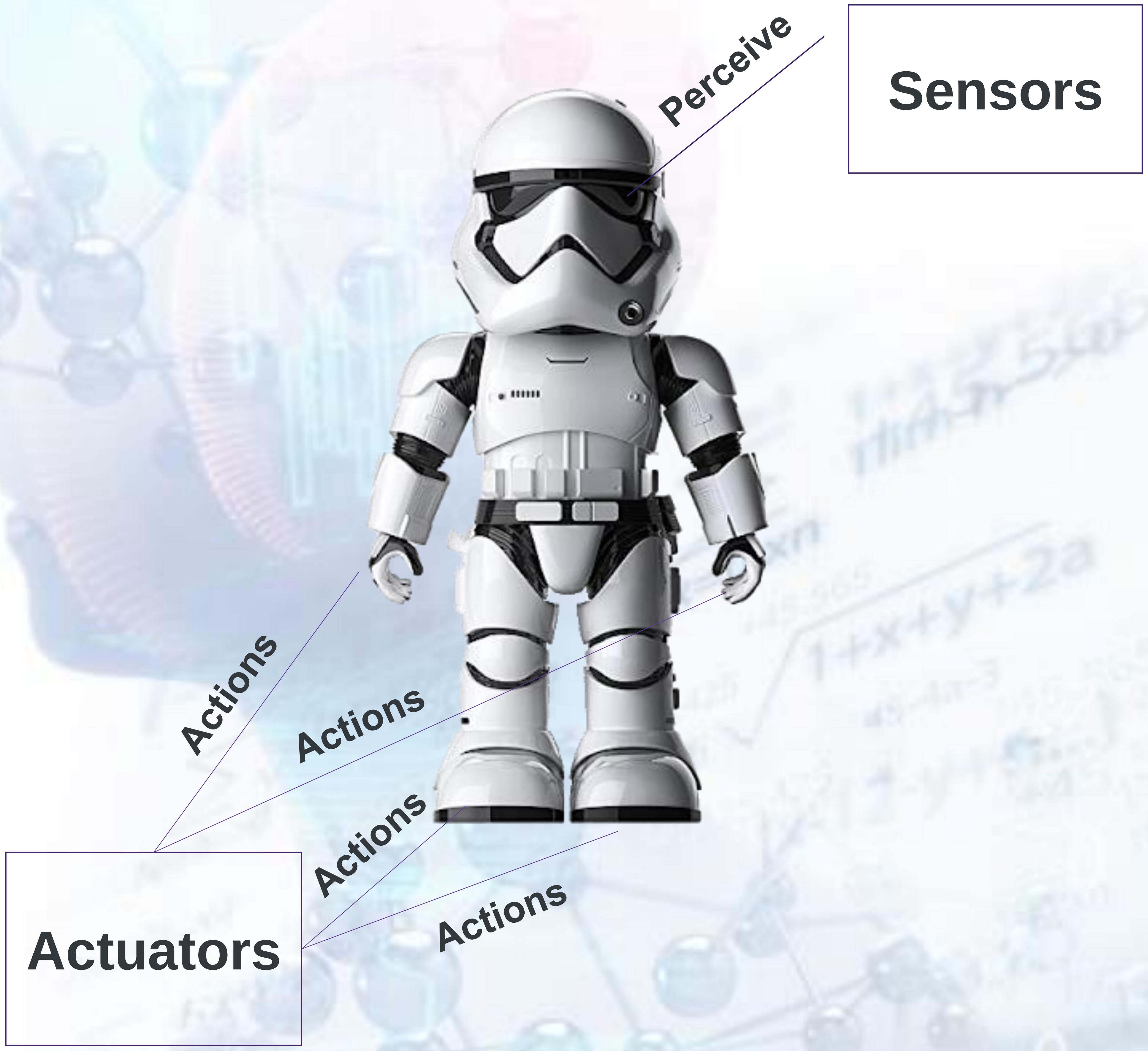
CSC-411 Artificial Intelligence

Intelligent Agents

Agents and Environments

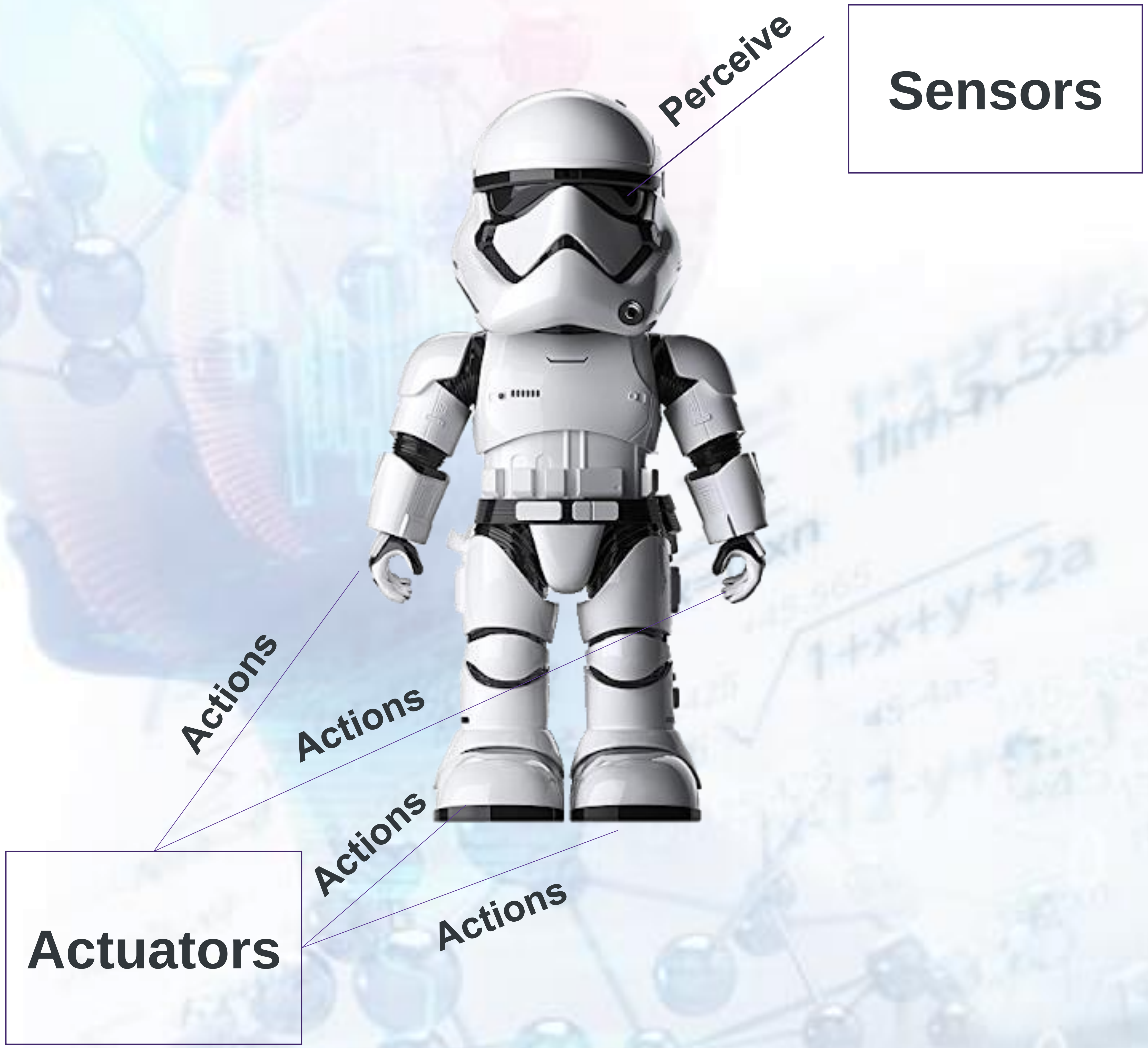
- Agent: An agent is anything that can be viewed as:
  - **Perceiving** its environment through **sensors** and
  - **Acting** upon that environment through **actuators**
- An agent program runs in cycles of:
  1. Perceive
  2. Think
  3. Act



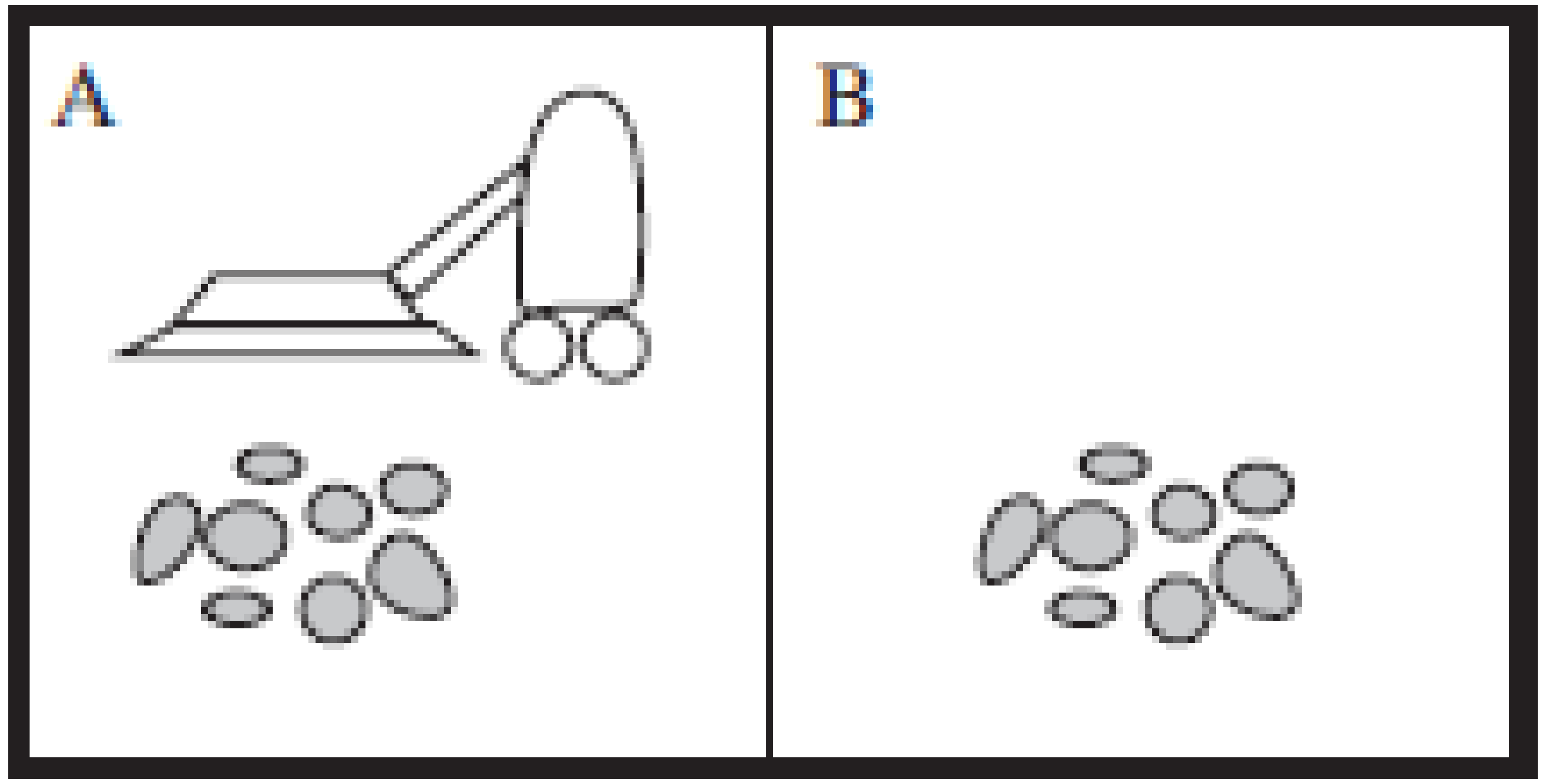
Agents and Environments

- Human agent:
  - Sensors: eyes, ears, and other organs.
  - Actuators: hands, legs, mouth, and other body parts.

- Robotic agent:
  - Sensors: Cameras and infrared range finders.
  - Actuators: Various motors.

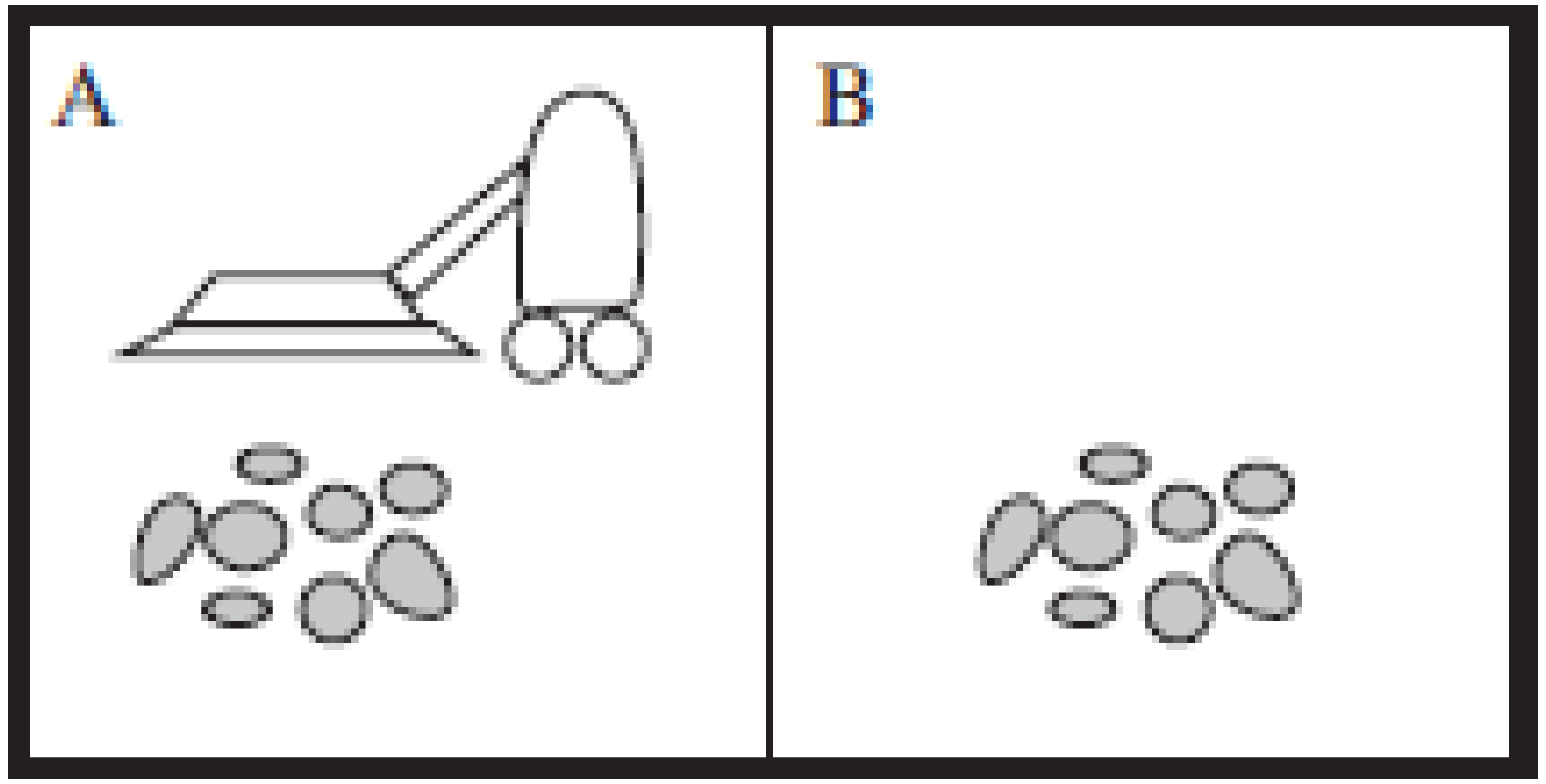


Intelligent Agent: Vacuum Cleaner



- **Percepts:** location and contents
  - [A, Dirty]
- **Actions:** Left, Right, Suck, NoOp
- **Agent function:** mapping from percepts to actions.

Intelligent Agent: Vacuum Cleaner



Percept	Action
[A, clean]	Right
[A, dirty]	Suck
[B, clean]	Left
[B, dirty]	Suck

Well-behaved Agents

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."

Rationality

- Rationality is relative to a performance measure.
- Judge rationality based on:
  - The performance measure that defines the criterion of success.
  - The agent’s prior knowledge of the environment.
  - The possible actions that the agent can perform.
  - The agent’s percept sequence to date.

PEAS

- When we define a rational agent, we group these properties under PEAS, the problem specification for the task environment.

- PEAS stands for:
  - Performance
  - Environment
  - Actuators
  - Sensors

PEAS

- What is **PEAS** for a self-driving car?
- **Performance**: Safety, time, legal drive, comfort.
- **Environment**: Roads, other cars, pedestrians, road signs.
- **Actuators**: Steering, accelerator, brake, signal, horn.
- **Sensors**: Camera, Sonar, GPS, Speedometer, Odometer, Accelerometer, Engine sensors, Keyboard.

Environment Types

- Fully Observable Environment
  - The Agent can observe **completely sufficient** information from the environment to make optimal decision.
- Partially Observable Environment
  - The Agent requires to **keep information in memory** to make an optimal decision.

Environment Types

- Deterministic Environment
  - Agent’s **actions** uniquely **determine** the **outcome**. (Chess)

- Stochastic Environment
  - The **outcome** of an **action** **cannot** be **predicted** **completely**. (Dice)

Environment Types

- Discrete Environment
  - **Finitely many** action **choices**, things to sense. (Chess)

- Continuous Environment
  - Space of **possible actions**, **things to sense** maybe **infinite**. (Darts)

Environment Types

- Benign Environment
  - Environment **might** be **random/stochastic**. (Weather)
- Adversarial Environment
  - Your **opponent** tries to **get you**. (Games)

Environment Types

- Single agent (vs. multi-agent):
  - An agent **operating by itself** in an environment.
  - If **multiple agents** are involved, **might collaborate** or **become adversaries**.

- Known (vs. Unknown):
  - The designer of the agent **may or may not** have **knowledge** about the **environment** makeup.
  - If the **environment** is **unknown** the agent will **need to know** how it works in order to decide.

Environment Types

	Partially Observable	Stochastic	Continuous	Adversarial
Checkers				✖
Poker	✖	✖		✖
Robotic Car	✖	✖	✖	

Types of Agent Programs

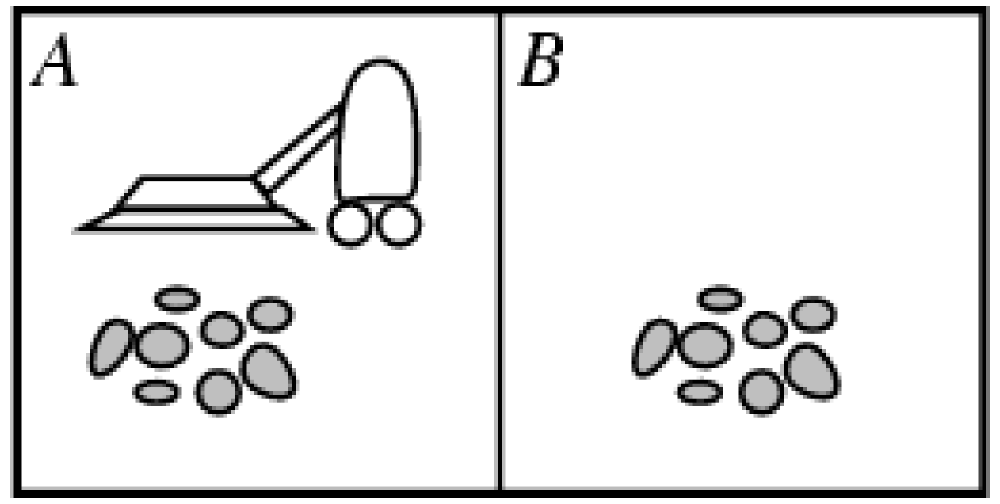
Types of Agent Programs

- Table-Lookup agents
- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents

Table-lookup Agents

- Uses a percept sequence-action table in memory to find the next action.
- Implemented as a (large) lookup table.
- Drawbacks:
  - Huge table (often simply too large)
  - Takes a long time to build/learn the table

Example: Vacuum World



- Percepts: robot senses it's location and "cleanliness."
  - So, location and contents, e.g., [A, Dirty], [B, Clean].
  - With 2 locations, we get 4 different possible sensor inputs.
- Actions: Left, Right, Suck, NoOp

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
….	

Table lookup Agents

- In more real-world scenarios, one would have many more different percepts (eg many more locations), e.g., >=100.
- There will therefore be 100^K different possible sequences of length K.
- For K = 20, this would require a table with over 100^20 entries!!
- So, table lookup formulation is mainly of theoretical interest. For practical agent systems, we need to find much more compact representations.
- For example, logic-based representations, Bayesian net representations, or neural net style representations, or use a different agent architecture,
- e.g., “ignore the past” --- Reflex agents.

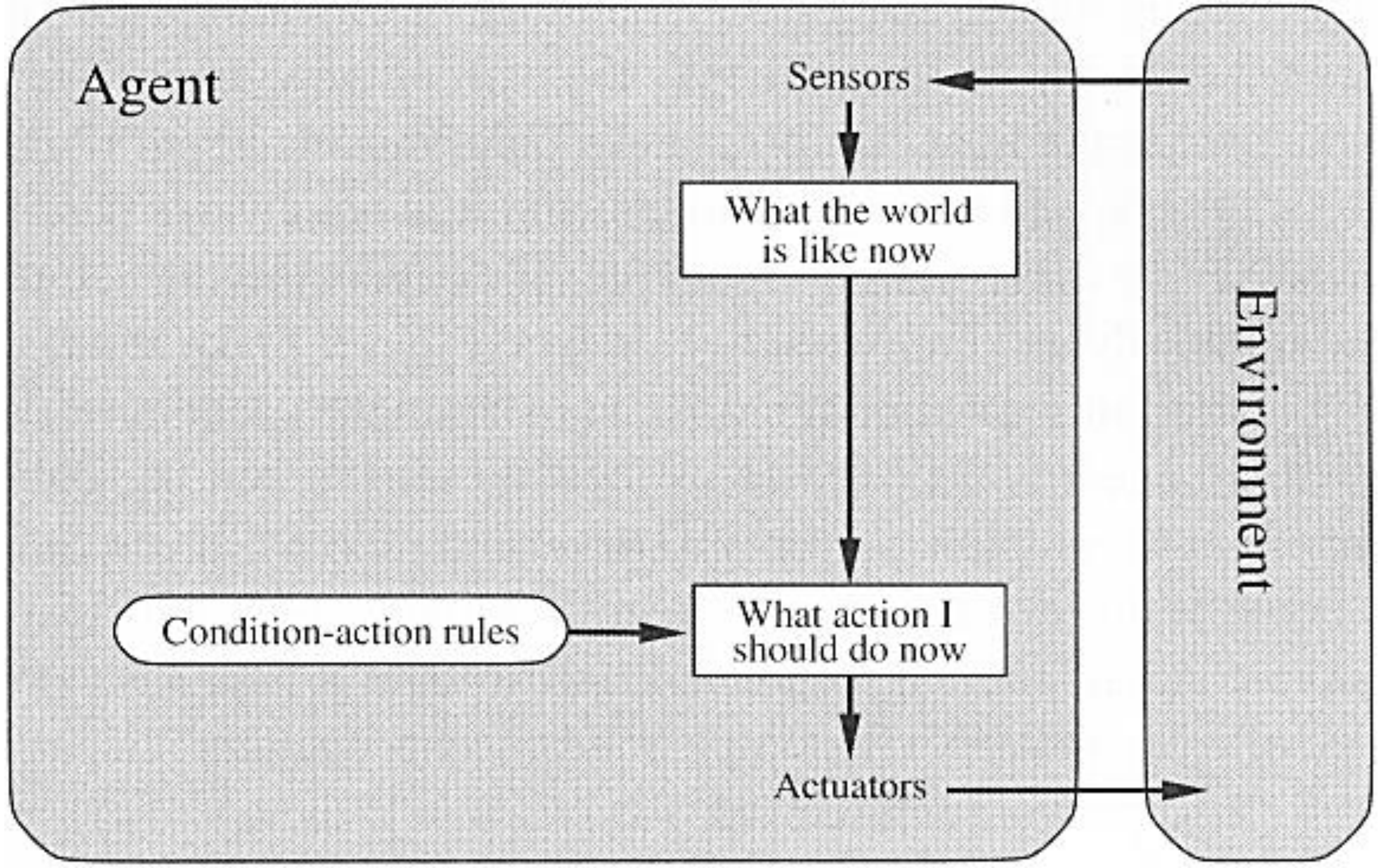
Simple Reflex Agents

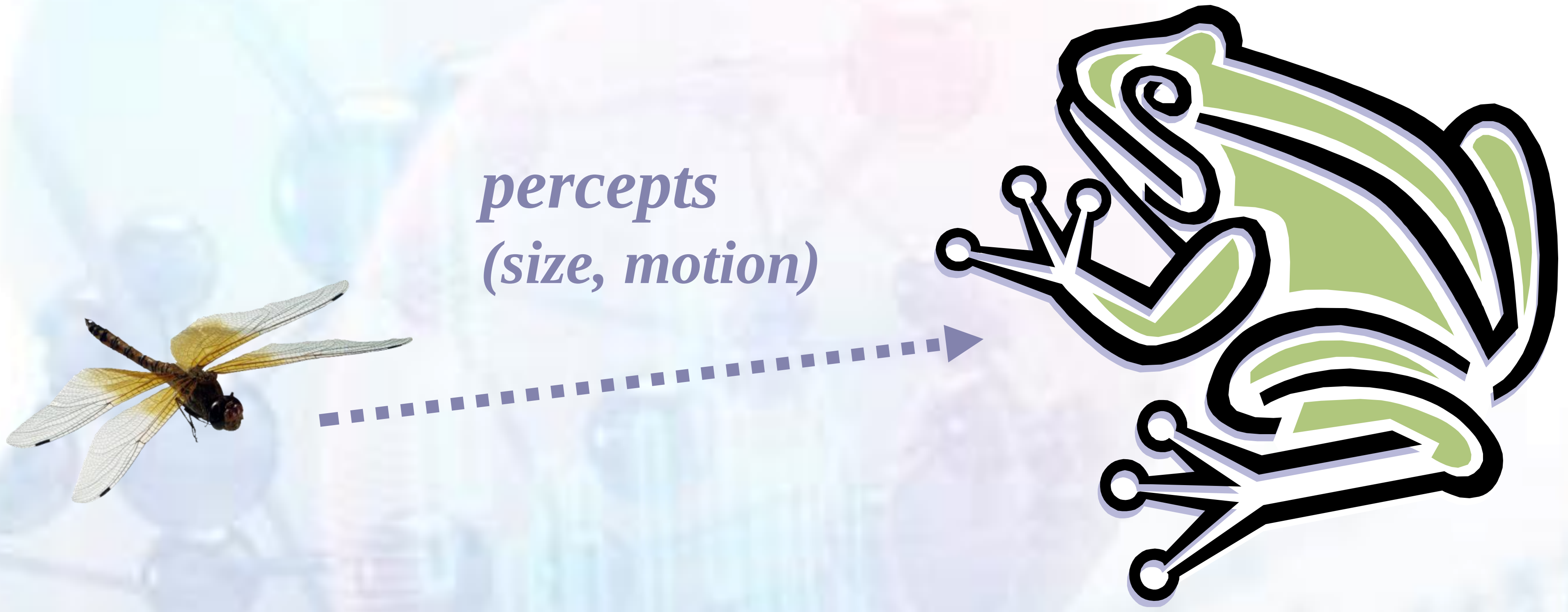
- It uses **condition-action rules**
  - The rules are like the form "if … then …"
  - efficient but have narrow range of applicability
  - Because knowledge sometimes cannot be stated explicitly
  - Work only if the environment is fully observable

Simple Reflex Agents

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

Simple Reflex Agents





RULES:

(1) If small moving object,
then activate SNAP

(2) (else) If large moving object,
then activate AVOID and inhibit SNAP

ELSE (not moving) then NOOP

Action: SNAP or AVOID or NOOP

Model-based Reflex Agents

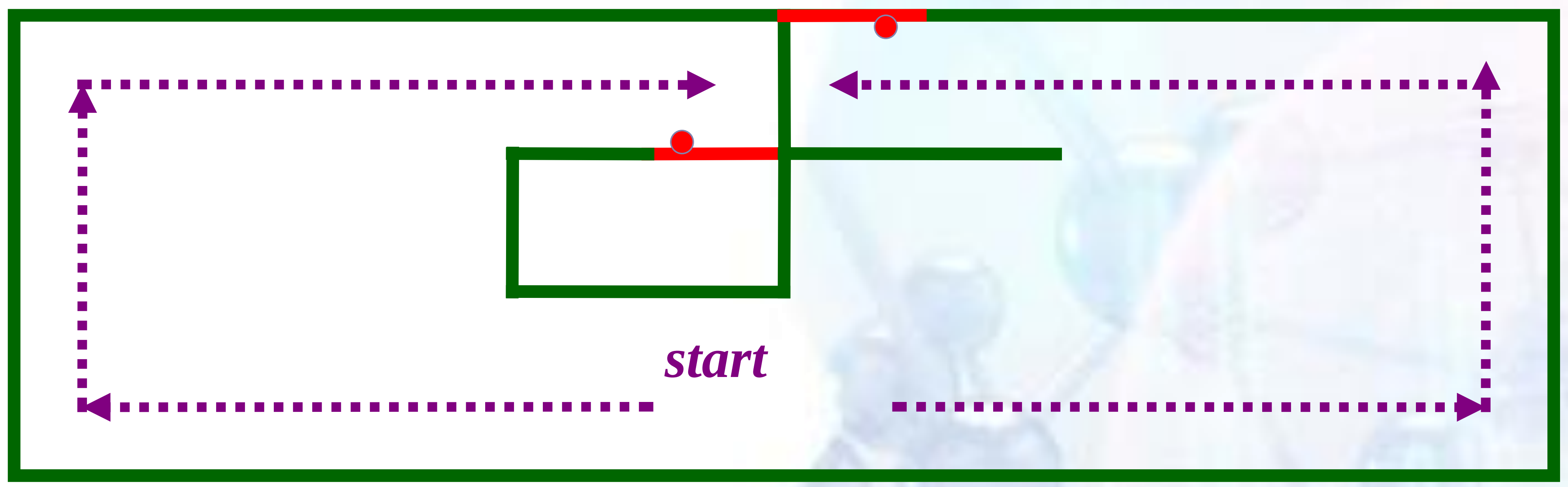
- For the world that is Partially observable
  - The agent has to keep track of an internal state:
    - That depends on the percept history
    - Reflecting some of the unobserved aspects
    - E.g., driving a car and changing lane

- Requiring 2 types of knowledge
  - How the world evolves independently of the agent
  - How the agent’s actions affect the world

Example Table Agent With Internal State

IF	THEN
Saw an object ahead, and turned right, and it’s now clear ahead	Go straight
Saw an object Ahead, turned right, and object ahead again	Halt
See no objects ahead	Go straight
See an object ahead	Turn randomly

Example Reflex Agent With Internal State: Wall-Following



Actions: left, right, straight, open-door

Rules:

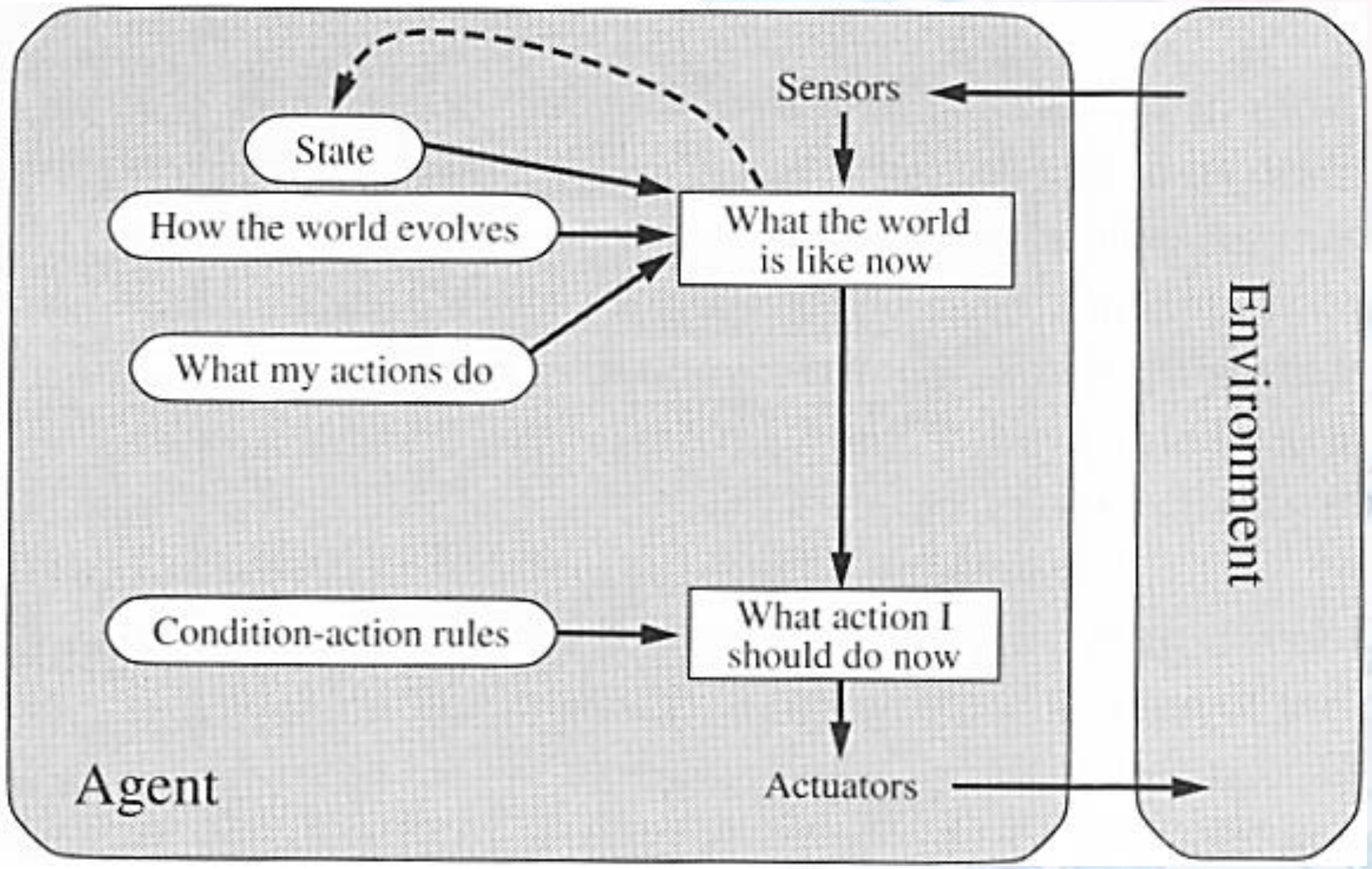
1. If open(left) & open(right) and open(straight) then choose randomly between right and left
2. If wall(left) and open(right) and open(straight) then straight
3. If wall(right) and open(left) and open(straight) then straight
4. If wall(right) and open(left) and wall(straight) then left
5. If wall(left) and open(right) and wall(straight) then right
6. If wall(left) and door(right) and wall(straight) then open-door
7. If wall(right) and wall(left) and open(straight) then straight.
8. (Default) Move randomly

Model-based Reflex Agents

function REFLEX-AGENT-WITH-STATE(percept) **returns** action
static: state, a description of the current world state
rules, a set of condition-action rules

state ← UPDATE-STATE(state,percept)
rule ← RULE-MATCH(state, rules)
action ← RULE-ACTION[rule]
state ← UPDATE-STATE(state, action)
return action

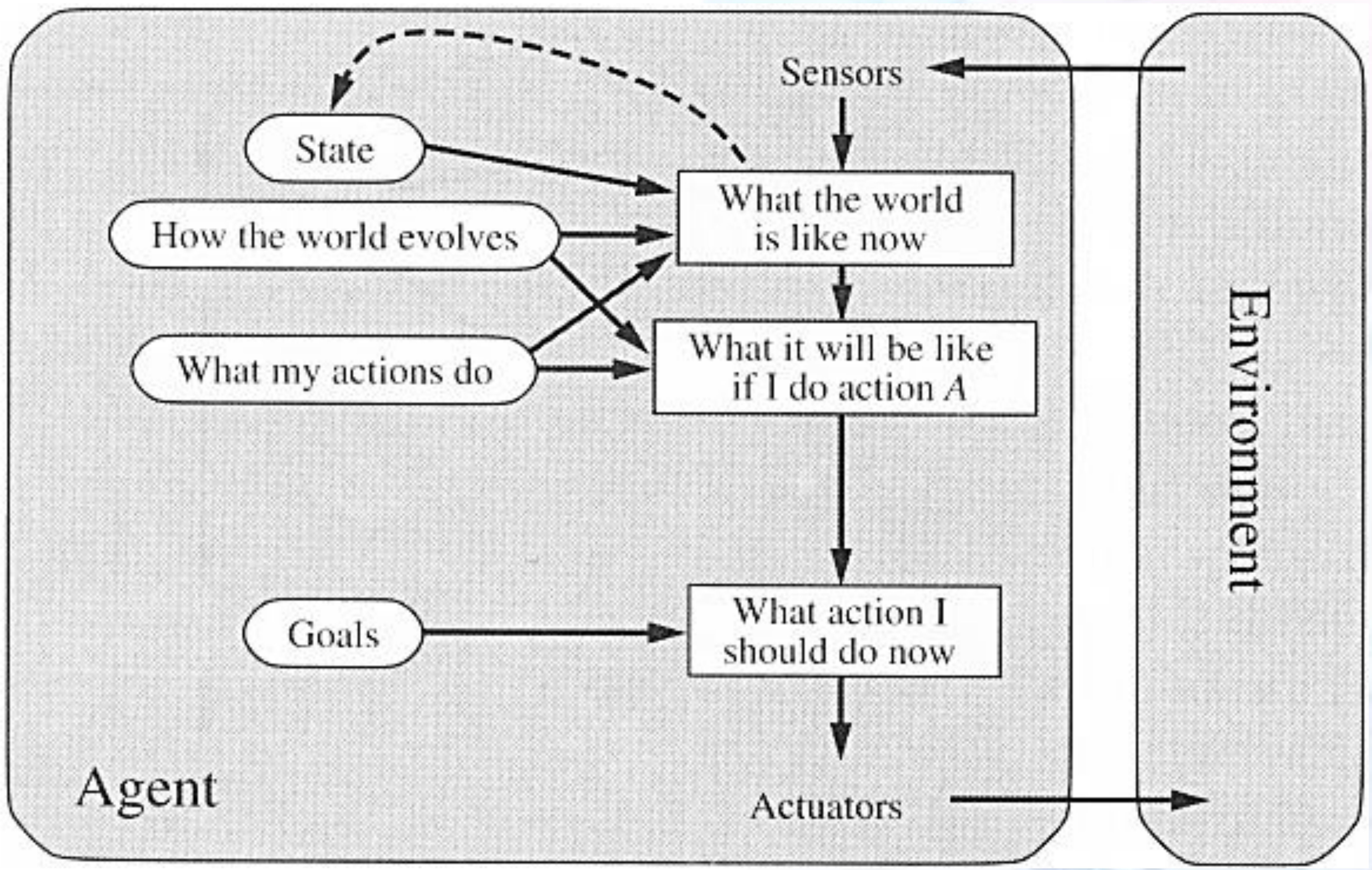
Model-based Reflex Agents



Goal-based Agents

- Current state of the environment is always not enough.
- The goal is another issue to achieve.
  - Judgment of rationality / correctness
- Actions chosen → goals, based on
  - The current state
  - The current percept

Goal-based Agents

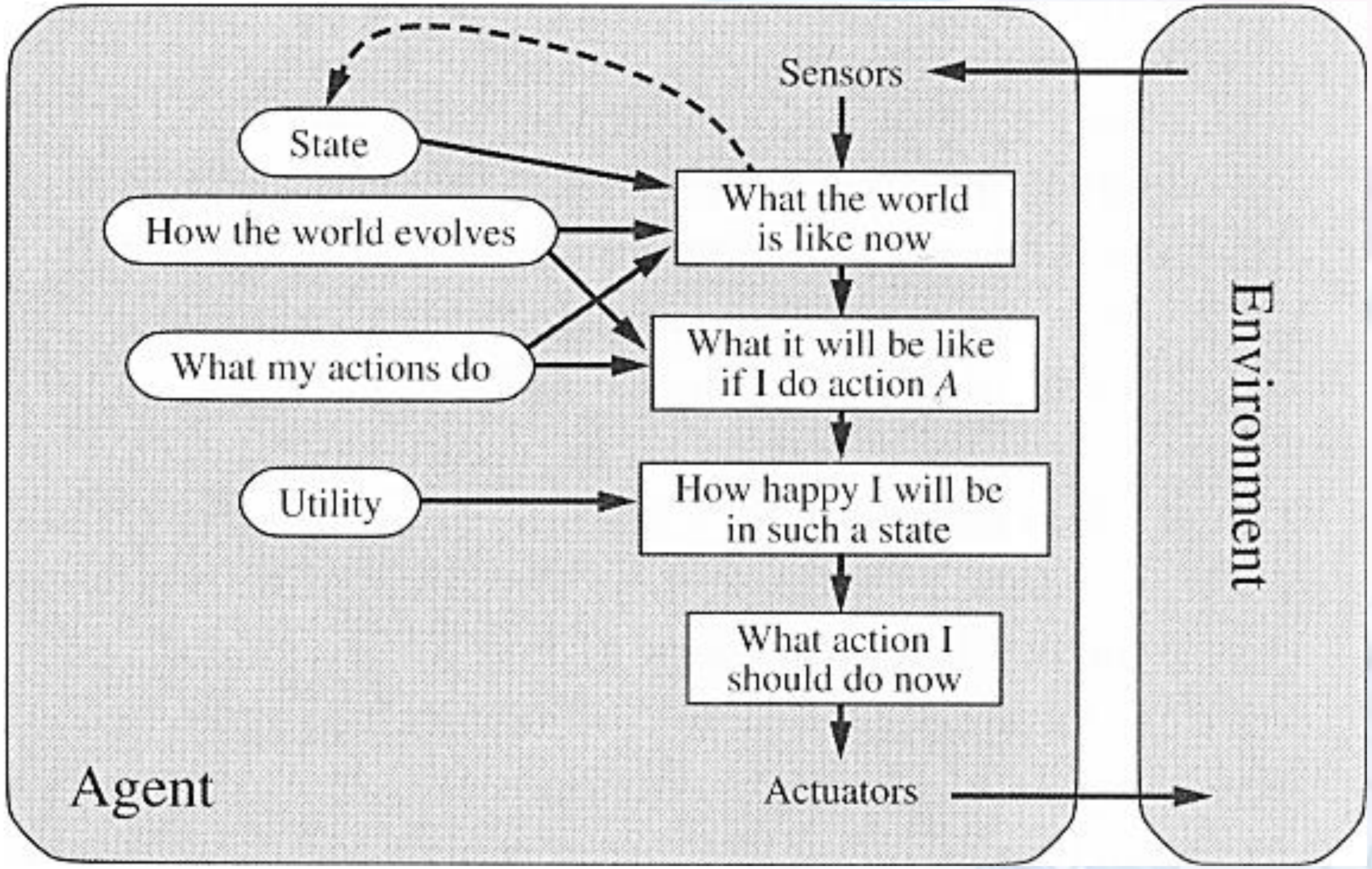


Utility-based Agents

- Goals alone are not enough
  - to generate high-quality behavior

- Many action sequences → the goals
  - Some are better and some worse
  - If goal means success,
    - Then utility means the degree of success (how successful it is)

Utility-based Agents



Utility-based Agents

- State A has higher utility
  - If state A is more preferred than others
- Utility is therefore a function
  - That maps a state onto a real number
  - The degree of success

Utility-based Agents

- Utility has several advantages:
  - When there are conflicting goals,
    - Only some of the goals but not all can be achieved
    - utility describes the appropriate trade-off
  - When there are several goals
    - None of them are achieved certainly
    - utility provides a way for the decision-making

Learning Agents

- Programming agents by hand can be very tedious.
- Four conceptual components:
  - Learning element: responsible for making improvements
  - Performance element: responsible for selecting external actions. It is what we considered as agent so far.
  - Critic: How well is the agent is doing w.r.t. a fixed performance standard.
  - Problem generator: allows the agent to explore.

Learning Agents