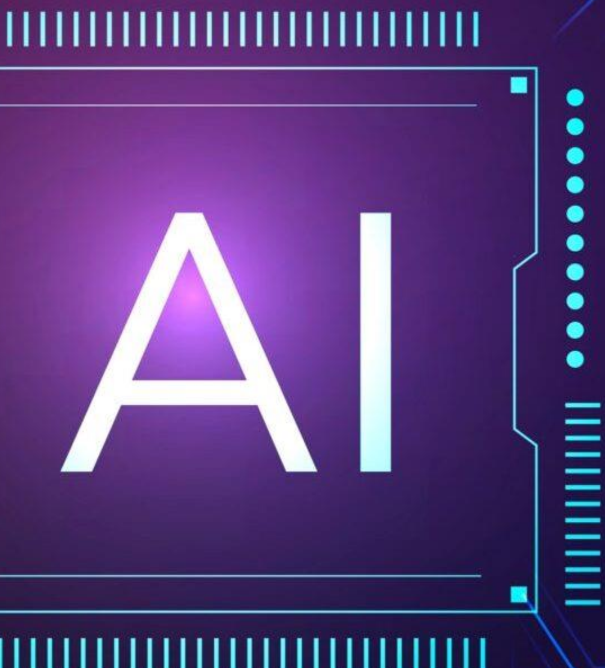


INTELLIGENT AGENTS

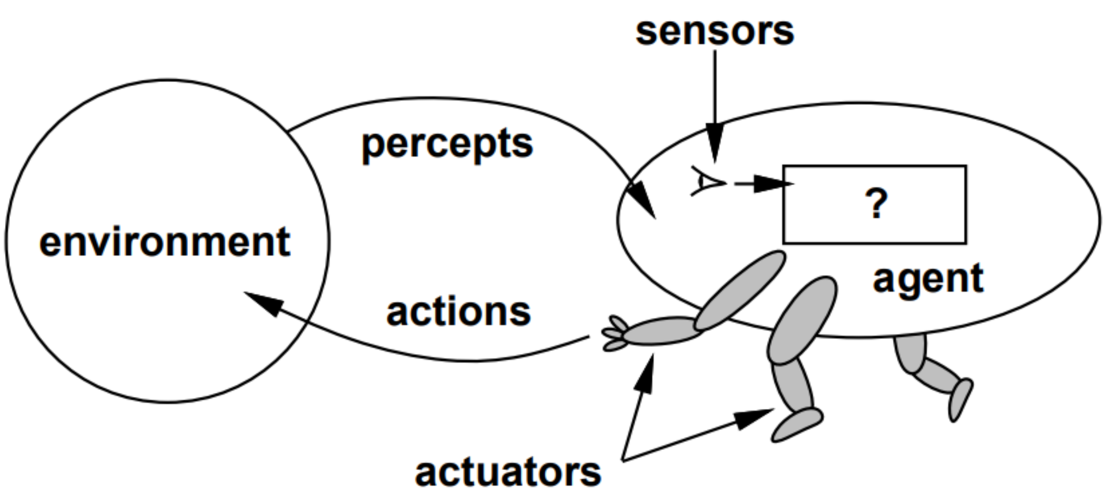
Nguyễn Ngọc Thảo – Nguyễn Hải Minh
{nnthao, nhminh}@fit.hcmus.edu.vn

Outline

- Agents and environments
- Good behavior: The concept of rationality
- The nature of environments
- The structure of agents (Self-study)

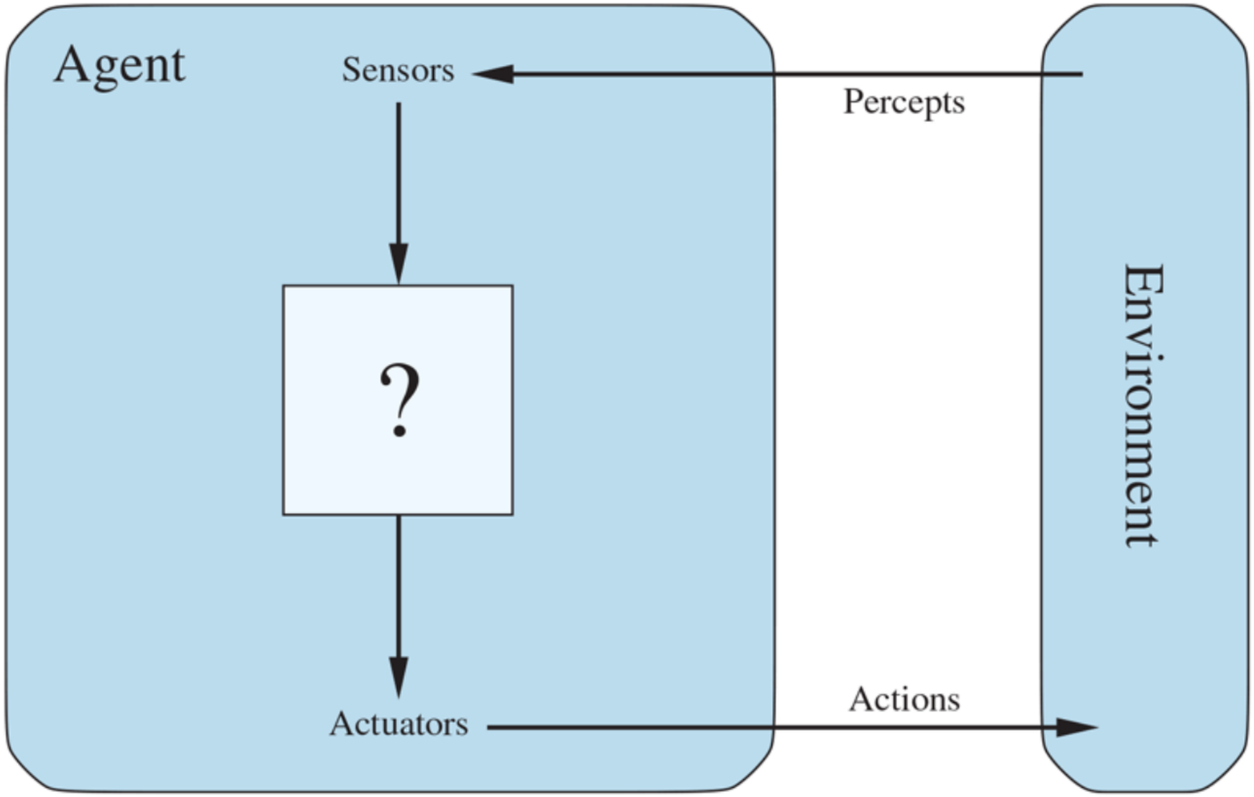
Agents and Environments

- *Agents: sensors and aactuators*
- *Agent function*
- *Agent program*



What is Agent?

- An agent perceives its environment through sensors and acts upon that environment through actuators.



Agents interact with environments through sensors and actuators.

Examples of agents

Human agent

Sensors: eyes, ears, and other organs.

Actuators: hands, legs, vocal tract, etc.

Robotic agent

Sensors: cameras, infrared range finders, etc.

Actuators: levels, motors, etc.

Software agent

Sensors: keystrokes, file contents, network packets, etc.

Actuators: monitor, physical disk, routers, etc.

The agent's behavior

- A percept refers to the content that an agent's sensors are perceiving.
- An agent's percept sequence is the complete history of everything the agent has ever perceived.

- An agent's choice of action at any given instant can depend on its built-in knowledge and on the entire percept sequence observed to date, but not on anything it hasn't perceived.

The agent's behavior

- An agent's behavior is described by the agent function that maps any given percept sequence to an action.

$$\boldsymbol{f}\colon\ \boldsymbol{\mathcal{P}} \rightarrow \boldsymbol{\mathcal{A}}$$

- This function can be described by tabulation.
    - A huge table is constructed by trying out all possible percept sequences and recording which actions the agent does in response.
- The table is an *external characterization* of the agent.

The agent's behavior

- The agent function for an artificial agent will be *internally implemented* by an agent program.

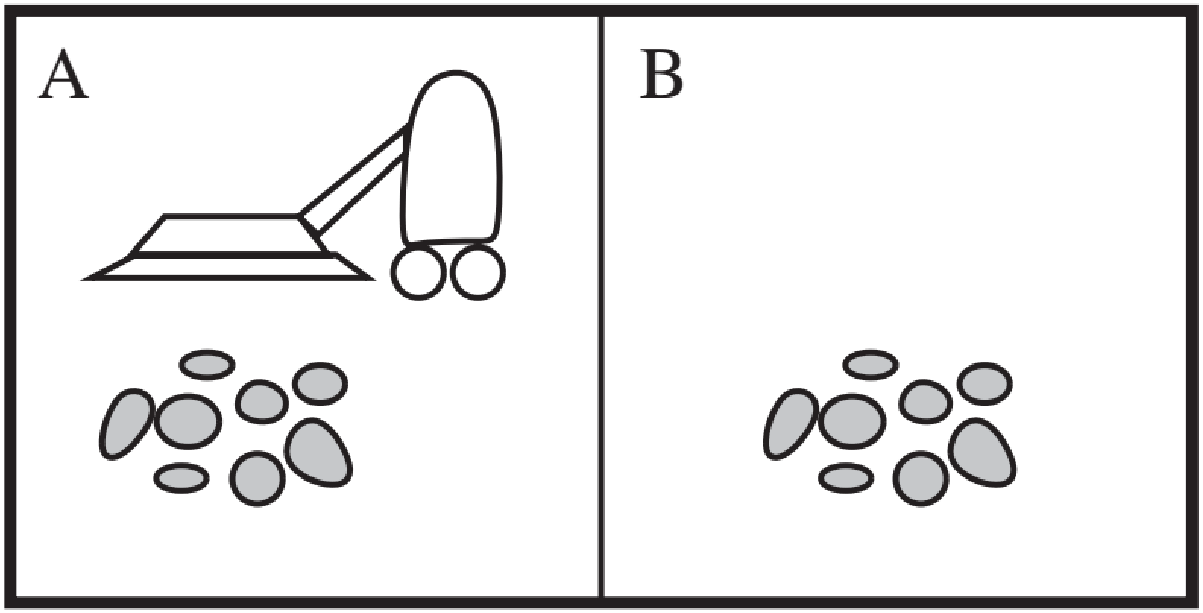
agent = architecture + program

mathematical

practical

The agent function is an abstract mathematical description;
The agent program is a concrete implementation, running within some physical system.

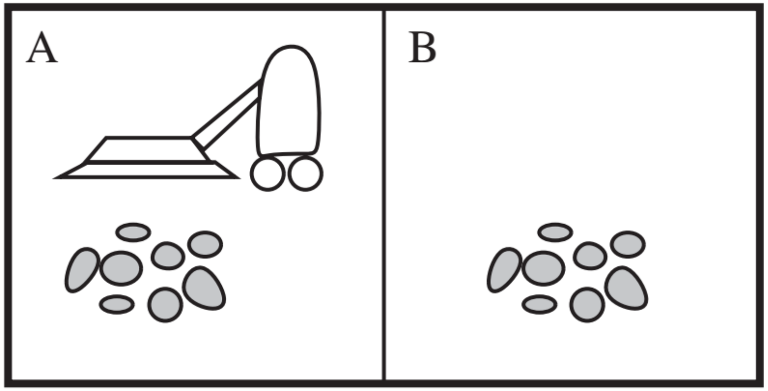
The Vacuum-cleaner world



A vacuum-cleaner world with just two locations

- Percepts: location and contents, e.g., [A,Dirty]
- Actions: Left, Right, Suck, Do Nothing

The Vacuum-cleaner world



Partial tabulation of a simple agent function for the vacuum-cleaner world

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
…	
…	
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck

function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action

if *status* = *Dirty* **then return** *Suck*

else if *location* = *A* **then return** *Right*

else if *location* = *B* **then return** *Left*

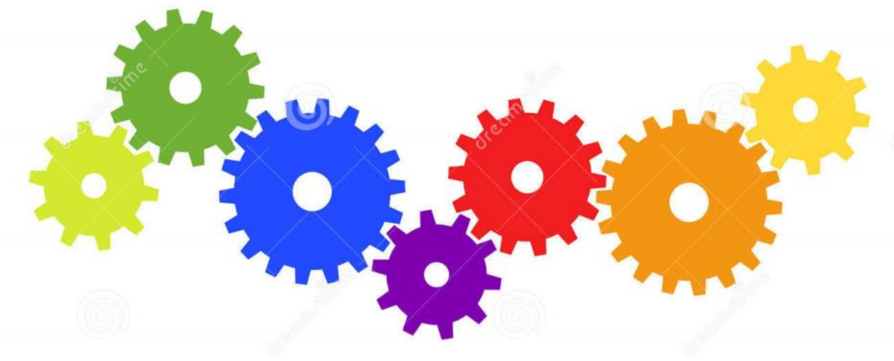
The agent program for a simple reflex agent in the two-state vacuum environment.

Why do we need agents?

- The notion of an **agent** is a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents.
- AI designs artifacts that have significant computational resources and the task environment requires nontrivial decision making.
  - E.g., a hand-held calculator can be viewed as an agent that displays “4” when given the percept sequence “2 + 2 =“ but this agent is too trivial for further study.

The concept of rationality

- *Rationality*
- *Omniscience, learning, and autonomy*



Rational agents

- A rational agent is one that does the right thing.
- But what does it mean to do the "right" thing?

- The actions that cause the agent to be most successful
  - E.g., Tabulation: every entry in the table for the agent function is filled out correctly.

- We need ways to measure success

Performance measure

Rational agents

- **Consequentialism:** we evaluate an agent's behavior by its consequences.

An agent, based on its percepts

⇩

generates actions sequence

⇩

environment goes to sequence of states

- If the sequence is desirable, the agent has performed well.

Performance measure

- Performance measure evaluates any given sequence of environment states (remember, not agent states!!!).
- **General rule:** Design performance measures according to

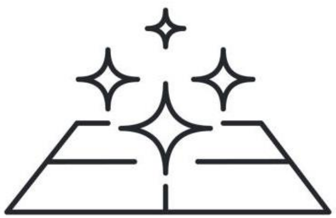
What one **actually wants** in the environment

Not how one thinks the agent should behave

- For example, in vacuum-cleaner world

The amount of dirt cleaned up in a single eight-hour shift

Which one is better?



The floor clean, no matter how the agent behaves

Rationality

- What is rational at any given time depends on

Performance measure Define the criterion of success	**Prior knowledge** What the agent knows about the environment
Percept sequence The agent's percept to date	**Actions** What the agent can perform

- Definition of a 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.*

The Vacuum-cleaner agent

- Performance measure: awards one point for each clean square at each time step, over a lifetime of 10000 timesteps
- Prior knowledge about the environment: The geography of the environment (2 squares) and the effect of the actions
- Actions that can perform: Left, Right, and Suck
- Percept sequences: The agent correctly perceives its location and whether that location contains dirt.
- Under this circumstance, the agent is **rational**.

The Vacuum-cleaner agent

- The same agent can be irrational under different scenarios.
- For example, the agent oscillate back and forth, even when all the dirt is cleaned up; and the performance measure includes a penalty of one point for each movement
- If the geography of the environment is unknown, the agent will need to explore it.

Omniscience vs. Rationality

- We need to distinguish between rationality and omniscience.

Omniscience	Rationality
• Know the actual outcome of actions in advance • No other possible outcomes • However, impossible in real world • Example?	Maximize performance measure given the percepts sequence to date and prior knowledge

Rationality is not perfection

Information gathering

- The agent must not engage in unintelligent activities due to **inadvertency**.
- Information gathering – Doing actions to modify future percepts (e.g., exploration)
- This is an important part of rationality.

Learning

- A rational agent not only gathers information but also **learns** as much as possible from what it perceives.
  - Its initial configuration may be modified and augmented as it gains experience.
- There are extreme cases in which the environment is completely known *a priori*.

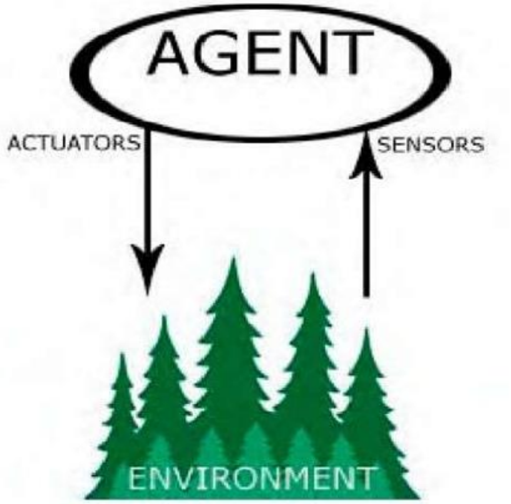
Autonomy

- A rational agent should be **autonomous** – Learn what it can to compensate for partial or incorrect prior knowledge.
- If an agent just relies on the prior knowledge of its designer rather than its own percepts, then the agent lacks autonomy.

A clock has no input (percepts) and run its own algorithm (prior knowledge). No learning, no experience, etc.

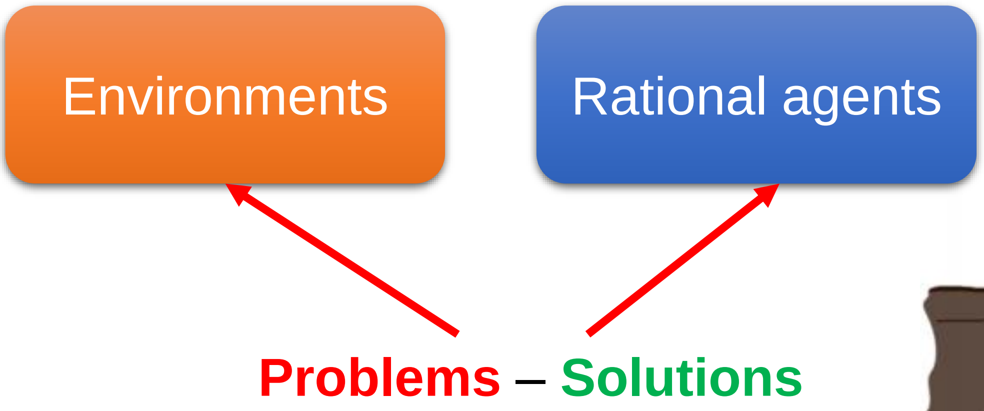
The nature of environments

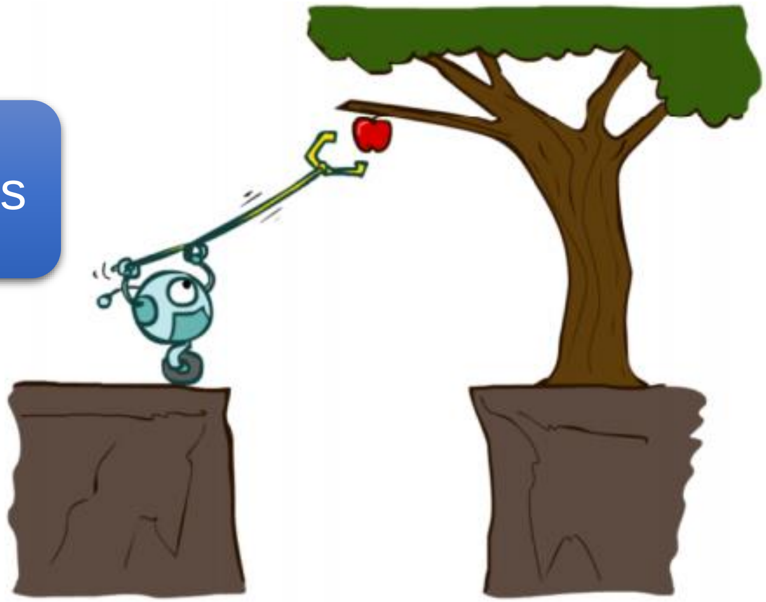
- *Specifying the task environment*
- *Properties of task environments*



The task environment

- **Task environments** are essentially the "problems" to which rational agents are the "solutions."





- They come in a variety of flavors, which directly affects the appropriate design for the agent program.

The task environment

- The task environment includes

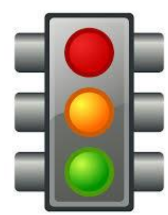
PEAS

- **P**erformance measure
- **E**nvironment
- Agent's **A**ctuators
- Agent's **S**ensors

- It must always be the first step in designing an agent and should be specified as fully as possible.

An example: Automated taxi driver

- Performance measure
  - How can we judge the automated driver?
  - Which factors are considered?
    - getting to the correct destination
    - minimizing fuel consumption
    - minimizing the trip time and/or cost
    - minimizing the violations of traffic laws
    - maximizing the safety and comfort
    - etc.



An example: Automated taxi driver

- Environment
  - A variety of roads (rural lane, urban alley, etc.)
  - Traffic lights, other vehicles, pedestrians, stray animals, road works, police cars, puddles, potholes, etc.
- Actuators (for outputs)
  - Control over the accelerator, steering, gear, shifting and braking
  - A display to interact with the passengers
- Sensors (for inputs)
  - Controllable cameras for detecting other vehicles, road situations
  - GPS (Global Positioning System) to know where the taxi is
  - More devices are necessary: speedometer, accelerometer, etc.

An example: Automated taxi driver

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

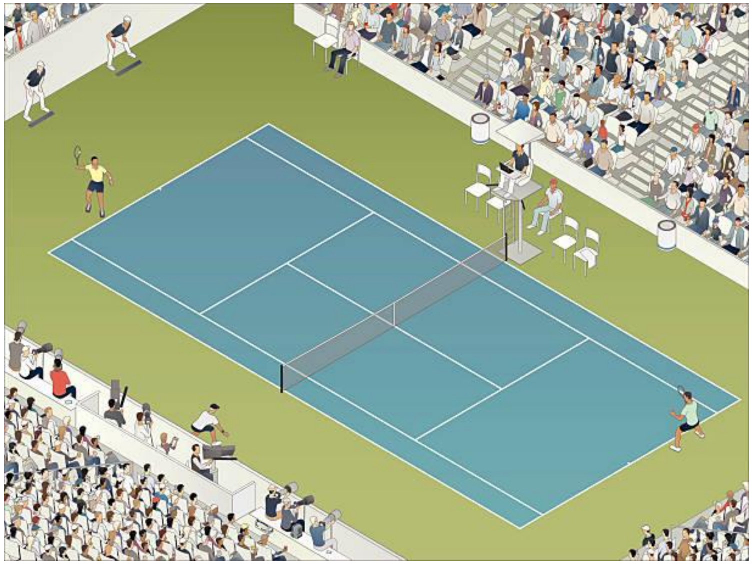
PEAS description of the task environment for an automated taxi.

Examples of agent types and their PEAS description

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Quiz 01: PEAS description

- For each of the following activities, give a PEAS description of the task environment



Playing a tennis match in a tournament

Practicing tennis against a wall

Properties of Task environment

- These dimensions determine the appropriate agent design and the applicability of techniques for agent implementation.

Fully observable	Partially observable
Single agent	Multiagent
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Known	Unknown

Fully Observable vs. Partially observable

- Fully observable: The agent's sensory gives it access to the complete state of the environment.
  - The agent need not maintain internal state to keep track of the world.
- Partially observable
  - Noisy and inaccurate sensors
  - Parts of the state are simply missing from the sensor data, e.g., a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares
- Unobservable: The agent has no sensors at all

Single agent vs. Multiagent

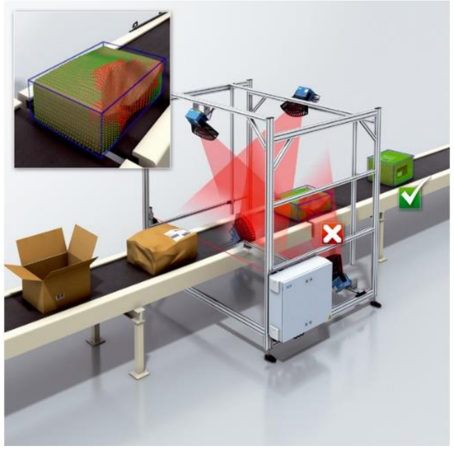
- Single agent: An agent operates by itself in an environment.
  - E.g., solving crossword $\rightarrow$ single-agent, playing chess $\rightarrow$ two-agent
- *Which entities must be viewed as agents?*
  - Whether B's behavior is described as maximizing a performance measure whose value depends on A's behavior.
- Competitive vs. Cooperative multiagent environment
  - E.g., playing chess $\rightarrow$ competitive, driving on road $\rightarrow$ cooperative

Deterministic vs. Stochastic

- Deterministic: The next state of the environment is completely determined by the current state and the action executed by the agent.
  - E.g., the vacuum world $\rightarrow$ deterministic, driving on road $\rightarrow$ stochastic
- Most real situations are so complex that they must be treated as stochastic.

Episodic vs. Sequential

- Episodic: The agent's experience is divided into atomic episodes, in each of which the agent receives a percept and then performs a single action.
  - Quality of action depends just on the episode itself
  - Do not need to think ahead
- Sequential: A current decision could affect future decisions.



Spotting defective parts on
an assembly line
vs.
Assembling a Lego house

Static vs. Dynamic

- Static: The environment is unchanged while an agent is deliberating.
  - E.g., crossword puzzles $\rightarrow$ static, taxi driving $\rightarrow$ dynamic
- Dynamic: The agent is continuously asked what it wants to do
  - If it has not decided yet, that counts as deciding to do nothing.

- Semi dynamic: The environment itself does not change with the passage of time but the agent's performance score does
  - E.g., chess playing with a clock

Properties of Task environment

- Discrete vs. continuous
  - The distinction applies to the state of the environment, to the way time is handled, and to the agent's percepts and actions
  - E.g., the chess has a finite number of distinct states, percepts and actions; while the vehicles' speeds and locations sweep through a range of continuous values smoothly over time.
- Known vs. unknown
  - Known environment: the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
  - Unknown environment: the agent needs to learn how it works to make good decisions.

Environments and their characteristics

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
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Examples of task environments and their characteristics.

Quiz 02: Task environment

- For each of the following activities, characterize its task environment in term of properties listed.
  - Playing a tennis match in a tournament
  - Practicing tennis against a wall

The structure of agents

- Simple reflex agent
- Model-based reflex agent
- Goal-based agents
- Utility-based agents

The agent architecture

agent = architecture + program

- Architecture: some sort of computing device with physical sensors and actuators that this program will run on.
  - Ordinary PC, robotic car with several onboard computers, cameras, and other sensors, etc.
- The program must be appropriate for the architecture.
  - Program: Walk action $\rightarrow$ Architecture: legs

The agent program

- An agent program takes the current percept from the sensors and returns an action to the actuators.
- The agent program takes only the current percept, because nothing more is available from the environment.
- Meanwhile, the agent function gets the entire percept sequence that the agent must remember.

A trivial agent program

- Keep track of the percept sequence and index into a table of actions to decide what to do.

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*

The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

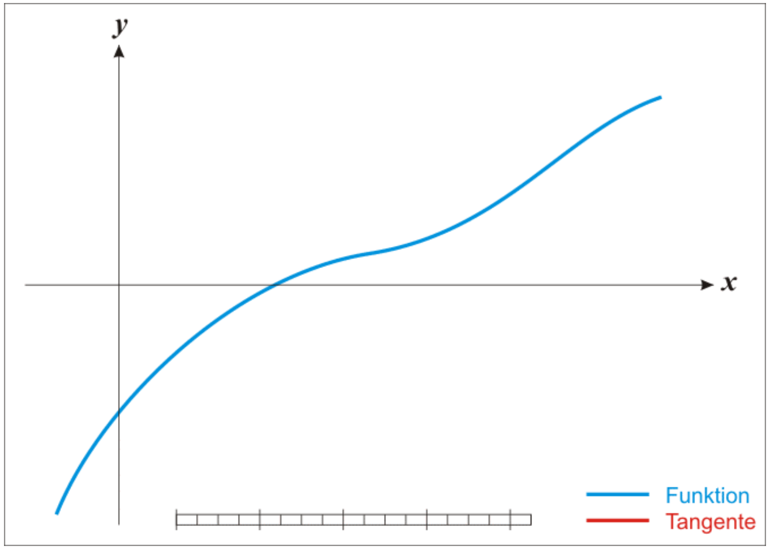
A trivial agent program

- $P =$ the set of possible percepts
- $T =$ lifetime of the agent
  - I.e., the total number of percepts it receives
- The size of the look up table is $\sum_{t=1}^{T} |P|^t$
- For example, consider playing chess
  - $P = 10,\ T = 150 \rightarrow$ A table of at least 10^{150} entries

- Despite of huge size, look up table does what we want

The key challenge of AI

- Write programs that produce rational behavior from a small amount of code rather than a large amount of table entries
  - E.g., calculate square roots – a five-line program of Newton's Method vs. a huge lookup tables



Source: [Wikipedia]

Squares - Cubes - Square Root Chart

number n	square n^2	cube n^3	square root $\sqrt{n}$	number n	square n^2	cube n^3	square root $\sqrt{n}$
1	1	1	1.0000	41	1681	68921	6.4031
2	4	8	1.4142	42	1764	74088	6.4807
3	9	27	1.7321	43	1849	79507	6.5574
4	16	64	2.0000	44	1936	85184	6.6332
5	25	125	2.2361	45	2025	91125	6.7082
6	36	216	2.4495	46	2116	97336	6.7823
7	49	343	2.6458	47	2209	103823	6.8557
8	64	512	2.8284	48	2304	110592	6.9282
9	81	729	3.0000	49	2401	117649	7.0000
10	100	1000	3.1623	50	2500	125000	7.0711
11	121	1331	3.3166	51	2601	132651	7.1414
12	144	1728	3.4641	52	2704	140608	7.2111
13	169	2197	3.6056	53	2809	148877	7.2801
14	196	2744	3.7417	54	2916	157464	7.3485
15	225	3375	3.8730	55	3025	166375	7.4162
16	256	4096	4.0000	56	3136	175616	7.4833
17	289	4913	4.1231	57	3249	185193	7.5498
18	324	5832	4.2426	58	3364	195112	7.6158
19	361	6859	4.3589	59	3481	205379	7.6811
20	400	8000	4.4721	60	3600	216000	7.7460
21	441	9261	4.5826	61	3721	226981	7.8102
22	484	10648	4.6904	62	3844	238328	7.8740
23	529	12167	4.7958	63	3969	250047	7.9373
24	576	13824	4.8990	64	4096	262144	8.0000
25	625	15625	5.0000	65	4225	274625	8.0623
26	676	17576	5.0990	66	4356	287496	8.1240
27	729	19683	5.1962	67	4489	300763	8.1854
28	784	21952	5.2915	68	4624	314432	8.2462
29	841	24389	5.3852	69	4761	328509	8.3066
30	900	27000	5.4772	70	4900	343000	8.3666
31	961	29791	5.5678	71	5041	357911	8.4261
32	1024	32768	5.6569	72	5184	373248	8.4853
33	1089	35937	5.7446	73	5329	389017	8.5440
34	1156	39304	5.8310	74	5476	405224	8.6023
35	1225	42875	5.9161	75	5625	421875	8.6603
36	1296	46656	6.0000	76	5776	438976	8.7178
37	1369	50653	6.0828	77	5929	456533	8.7750
38	1444	54872	6.1644	78	6084	474552	8.8318
39	1521	59319	6.2450	79	6241	493039	8.8882
40	1600	64000	6.3246	80	6400	512000	8.9443

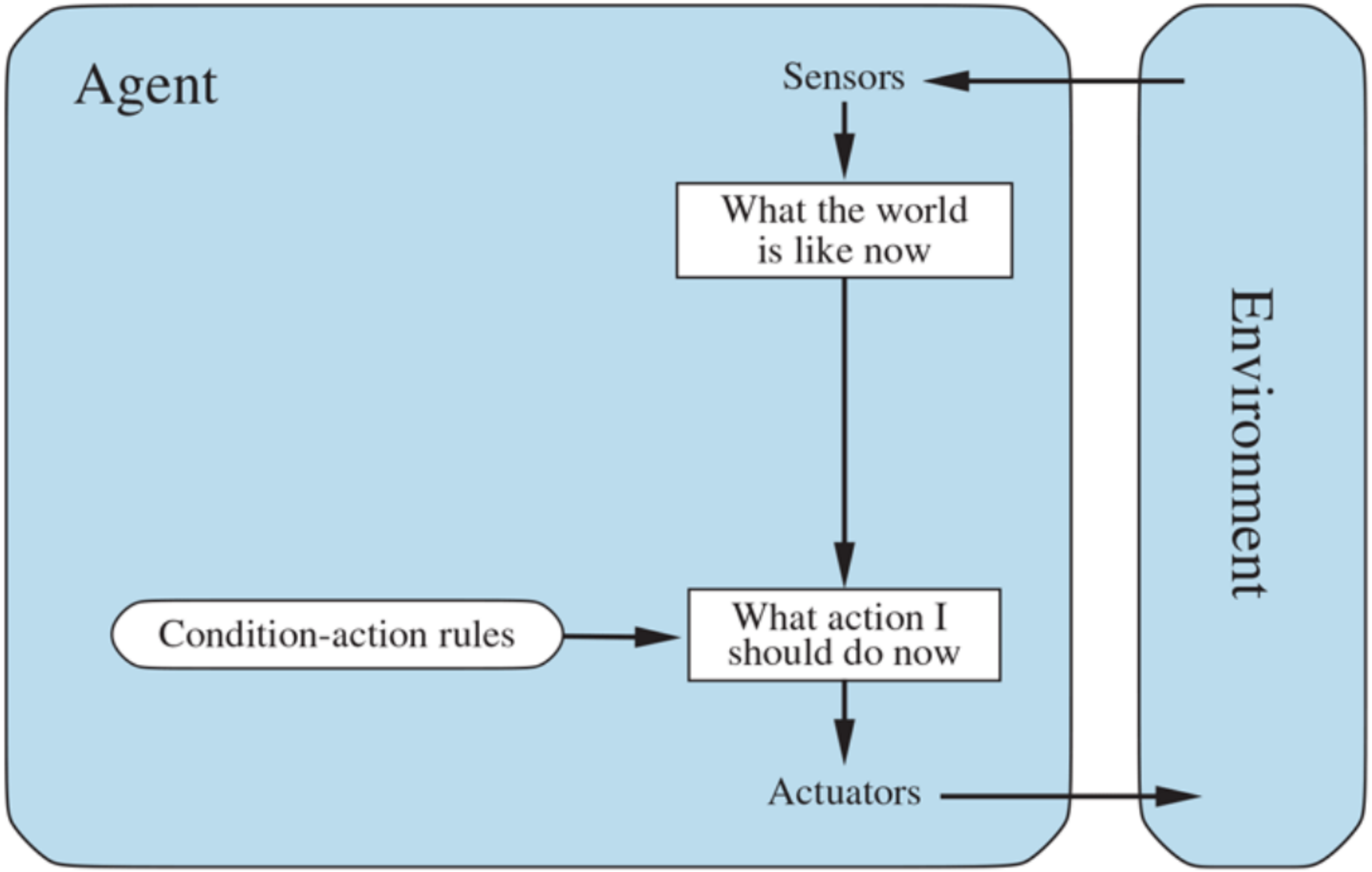
Simple reflex agents

- The simplest kind of agent, limited intelligence
- Select actions based on the current percept, ignoring the rest of the percept history
- The connection from percept to action is represented by condition-action rules.

IF *current percept* **THEN** *action*

  - E.g., IF *car-in-front-is-braking* THEN *initiate-braking*.

- Limitations
  - Knowledge sometimes cannot be stated explicitly → low applicability
  - Work only if the environment is fully observable



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*)

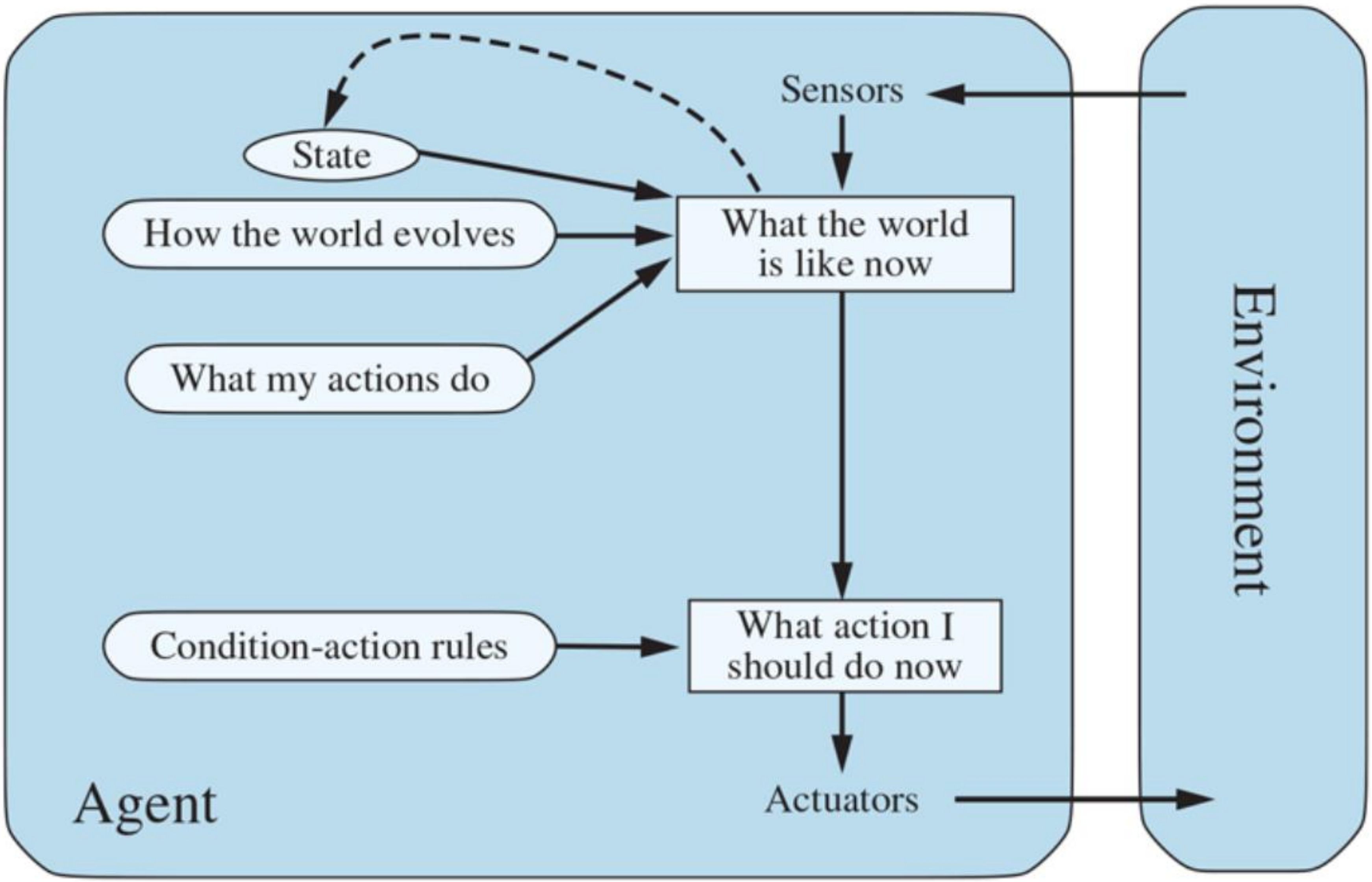
action ← *rule*.ACTION

return action

Model-based reflex agents

- The agent must keep track of an internal state in partially observable environments.
- This state depends on the percept history and reflects some of the unobserved aspects.
- The agent program updates the internal state information as time goes by by encoding two kinds of knowledge

Transition model	How the world works
Sensor model	How the state of the world is reflected in the agent's percepts


Sensors
State
How the world evolves
What the world is like now
What my actions do
Environment
Condition-action rules
What action I should do now
Agent
Actuators

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action

persistent: *state,* the agent's current conception of the word state

transition_model, a description of how the next state depends on the current state and action

sensor_model, a description of ho the current world state is reflected in the agent's percepts

rules, a set of condition–action rules

action, the most recent action, initially none

state ← UPDATE-STATE(*state, action, percept, transition_model, sensor_model*)

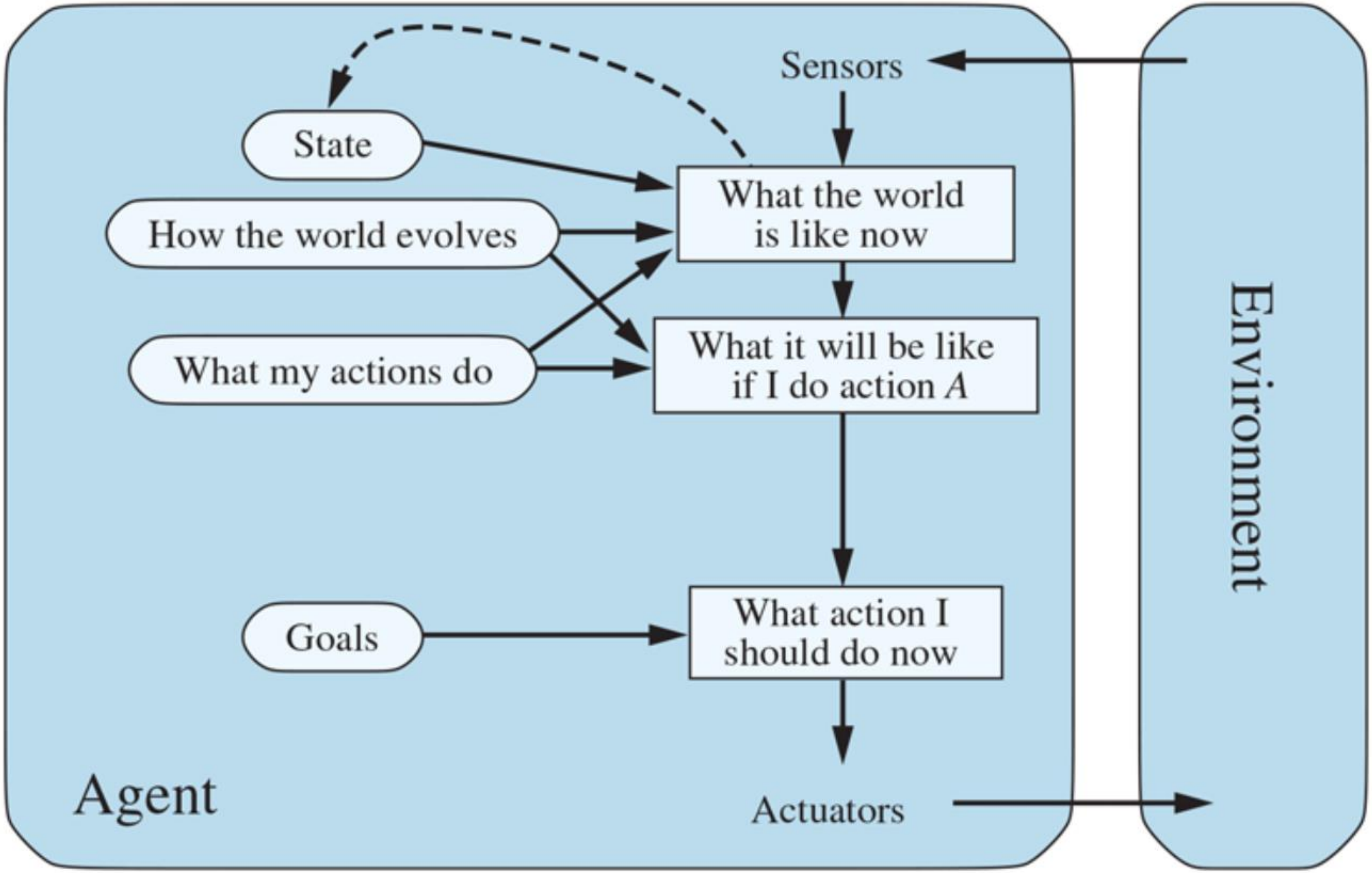
rule ← RULE-MATCH(*state*, *rules*)

action ← *rule*.ACTION

return action

Goal-based agents

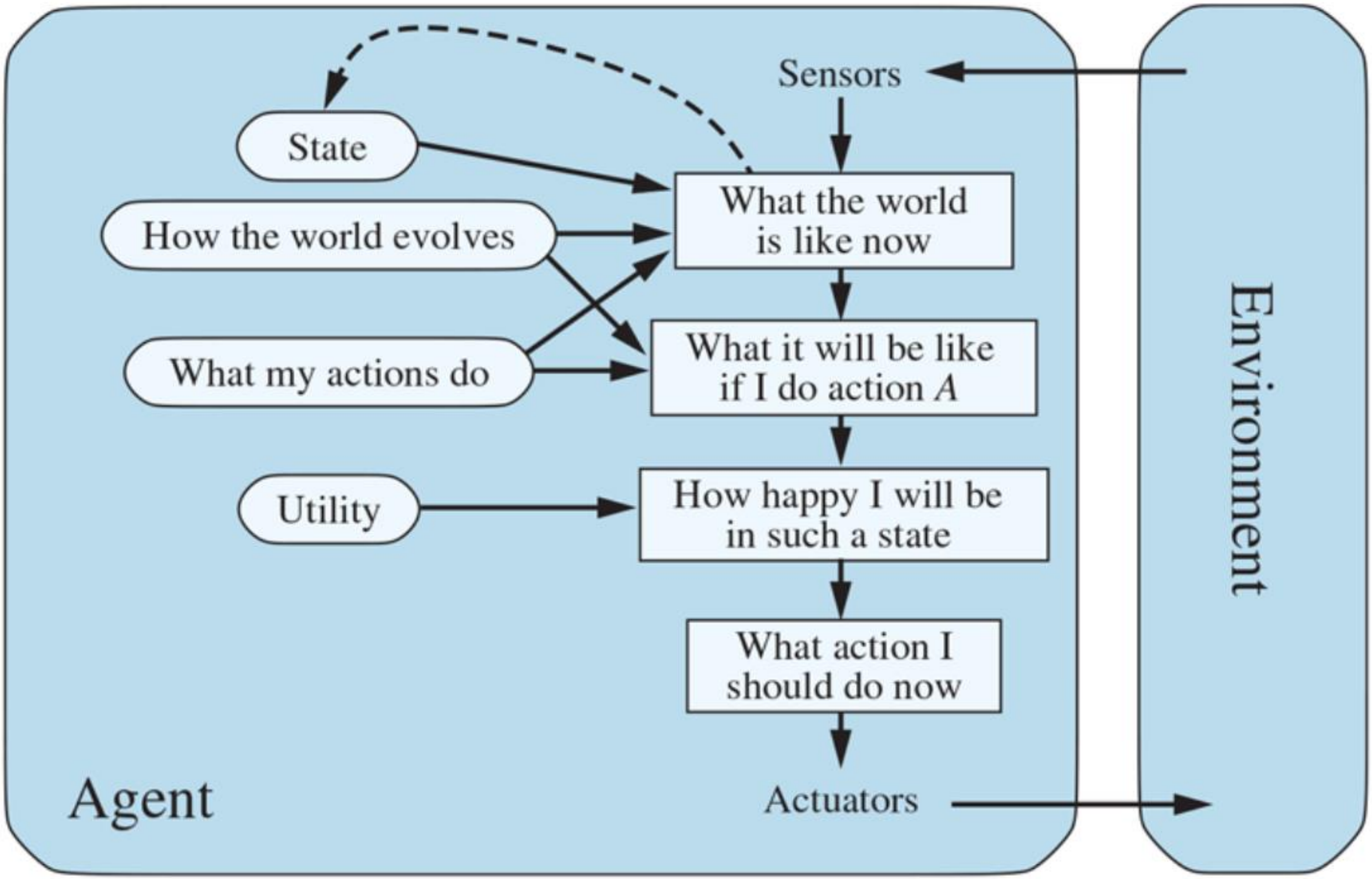
- Current state of the environment is always not enough
- The agent further needs some sort of goal information that describes desired situations.
  - E.g., at a road junction, the taxi can turn left, turn right, or go straight on, depending on where the taxi is trying to get to.
- Less efficient but more flexible
  - Knowledge supporting the 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 the achievement of its goals.

Utility-based agent

- Goals are inadequate to generate high-quality behavior in most environments.
  - Many action sequences can get the goals, some are better, and some are worse, e.g., go home by taxi or Grab car?
- An agent’s utility function is essentially an internalization of the performance measure.
  - Goal $\rightarrow$ success, utility $\rightarrow$ degree of success (how successful it is)
  - If state A is more preferred than others, then A has higher utility.



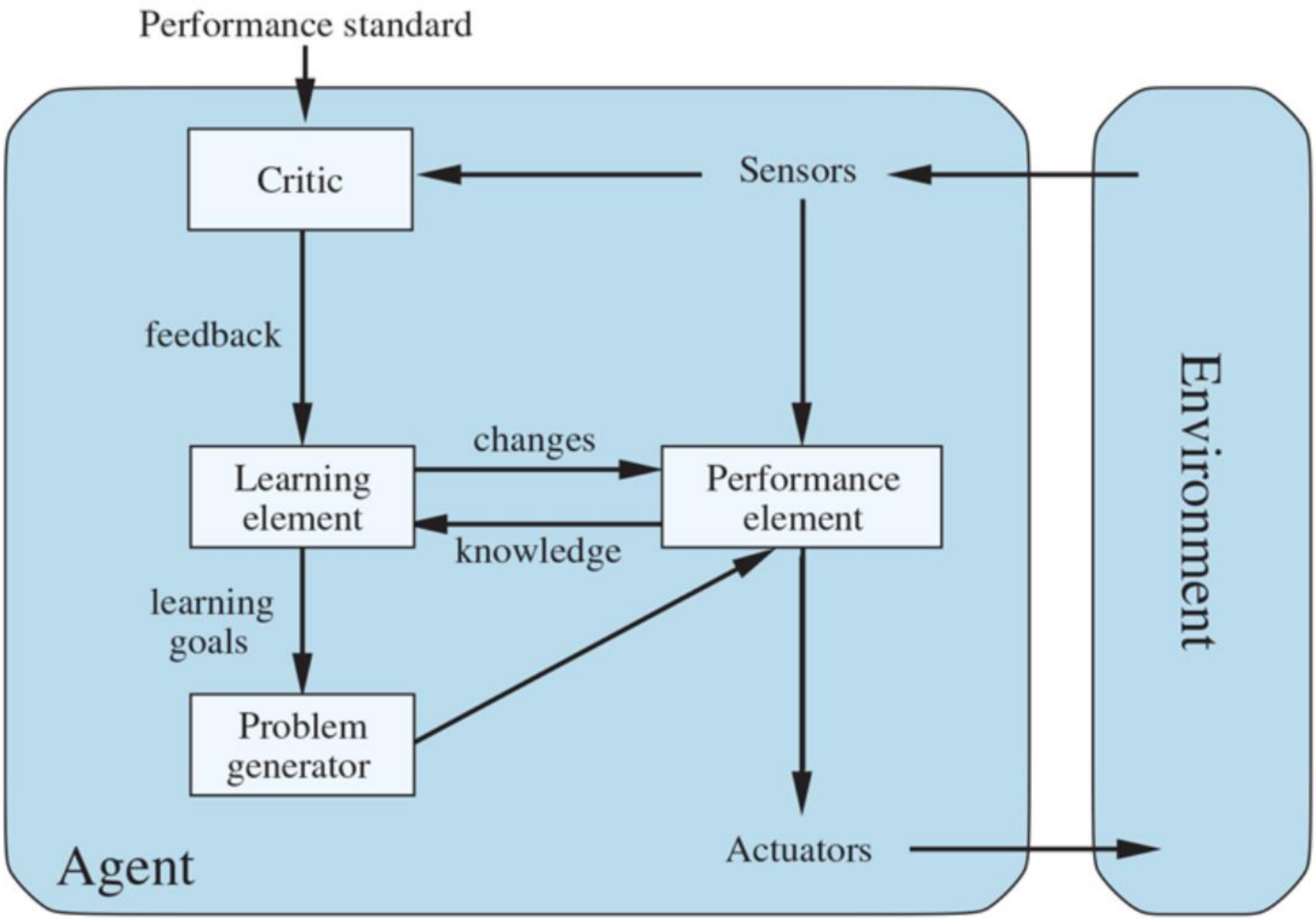
A utility-based uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Utility-based agent: Advantages

- When there are conflicting goals
  - Only some of which can be achieved, e.g., speed and safety
  - The utility function specifies the appropriate tradeoff.
- When there are several goals that the agent can aim for
  - None of which can be achieved with certainty
  - The utility weights the likelihood of success against the importance of the goals.
- The rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes

Learning agents

- After an agent is programmed, can it work immediately?
  - No, it still need teaching
- Once an agent is done, what can we do next?
  - Teach it by giving it a set of examples
  - Test it by using another set of examples
- We then say the agent **learns** $\rightarrow$ learning agents



A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

Conceptual components in learning agents

1. Learning element $\rightarrow$ Make improvement
2. Performance element $\rightarrow$ Select external actions
3. Critic $\rightarrow$ Tell the Learning element how well the agent is doing with respect to fixed performance standard. (Feedback from user or examples, good or not?)
4. Problem generator $\rightarrow$ Suggest actions leading to new and informative experiences

How learning agents learn?

- It is a process of modifying each component of the agent
- To bring the components into closer agreement with the available feedback information
- Thereby, improve the overall performance of the agent.

Learning agents: An example

- Performance element
  - Whatever collection of knowledge and procedures the taxi has for selecting its driving actions (may be further modified)
- Critic
  - Observe the world and pass information to the learning element
  - E.g., quick left turn across three lanes of traffic $\rightarrow$ shocking language used by other drivers observed $\rightarrow$ bad action
- Learning element
  - Formulate new rules from the experience told by the critic
  - E.g., a new rule for the above bad action
- Problem generator
  - Identify certain behaviors in need of improvement and suggest experiments
  - E.g., try out the brakes on different road surfaces under different conditions

Quiz 03: Learning agents

- Give an example of learning rational agent following four conceptual elements.


...the end.