

CS 480

Introduction to Artificial Intelligence

January 16, 2024

Announcements / Reminders

- **Contribute to the discussion on Blackboard if you can, please**
- **Please follow the Week 01 To Do List instructions (if you haven't already):**
 - **Go through the Syllabus,**
 - **Setup Python environment on your computer**
 - **READ the assigned material**

Teaching Assistants

Name	e-mail	Office hours
Donthula, Sumanth	sdonthula@hawk.iit.edu	TBD
Nagaraju, Ashish	anagaraju@hawk.iit.edu	TBD

TAs will:

- assist you with your assignments,
- hold office hours to answer your questions,
- grade your lab work (**a specific TA will be assigned to you**).

Take advantage of their time and knowledge!

DO NOT email them with questions unrelated to lab grading.

Make time to meet them during their office hours.

Add a [CS480 Spring 2024] prefix to your email subject when contacting TAs, please.

Plan for Today


- **Intelligent Agents**

Identifying Problems Suitable for AI

Most AI problems will exhibit the following three characteristics:

- tend to be large,
- computationally complex and cannot be solved by a straightforward algorithm,
- tend to require a significant amount of human expertise to be solved

Agent-Based Modeling



Pandemic influenza computer model (no soundtrack)

Los Alamos National Lab
16.2K subscribers

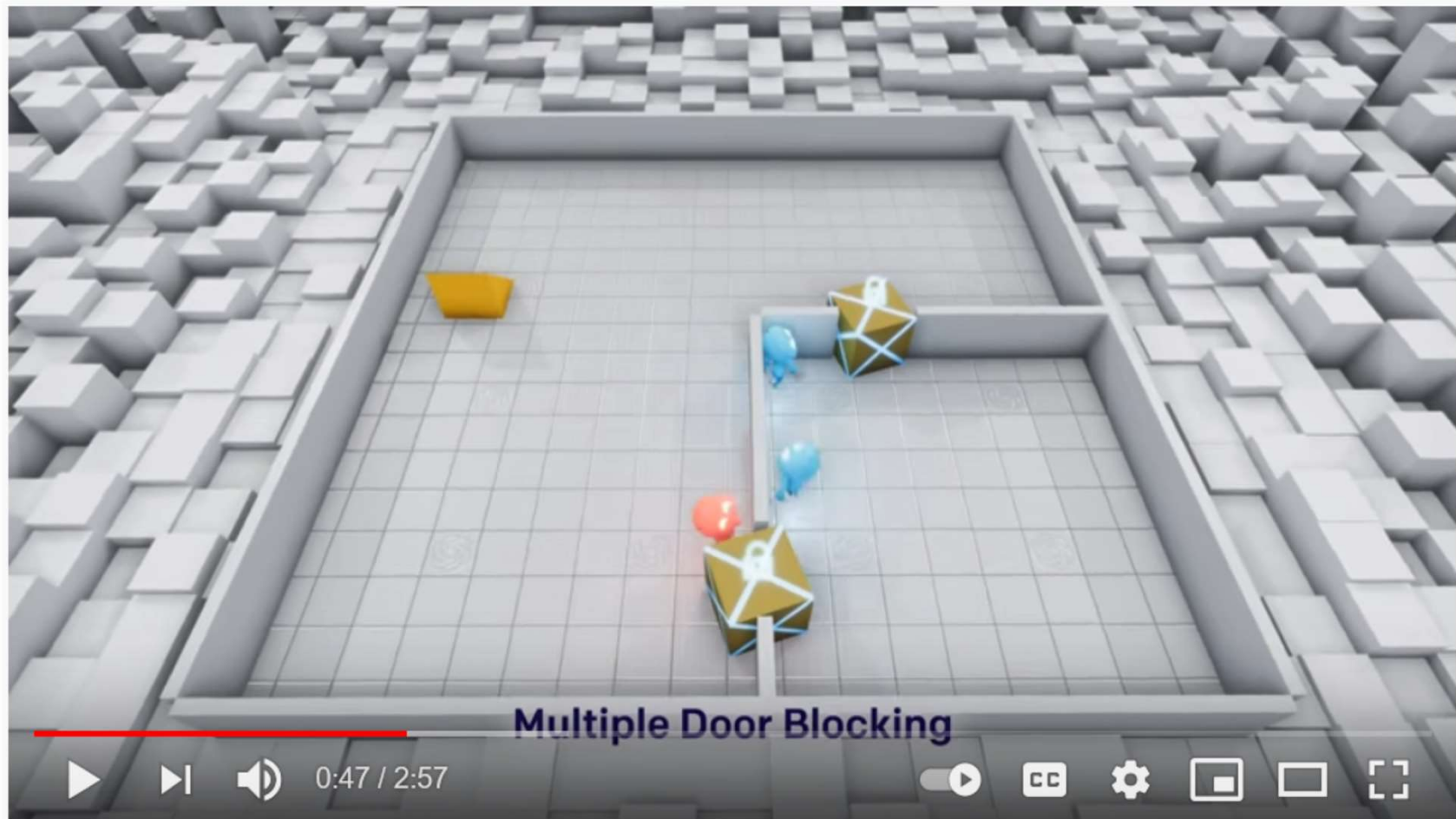
Subscribe

24 | Share | Save

Source: https://www.youtube.com/watch?v=E_-9hFzmxkw

Intelligent (Autonomous) Agents

Intelligent Agents in Action



Multi-Agent Hide and Seek

4,588,797 views • Sep 17, 2019



120K



1.7K



SHARE



SAVE

...

Source: <https://www.youtube.com/watch?v=kopoLzvh5jY>

Agent

Agent:

An **agent** is just **something that acts** (from the Latin agere, to do).

Of course, we would prefer “acting” to be:

- autonomous
- situated in some environment (that could be really complex)
- adaptive
- creative and goal-oriented

Rational Agent

Rational Agent:

A **rational agent** is one that acts so as **to achieve the best outcome**, or when there is uncertainty, **the best expected outcome**.*

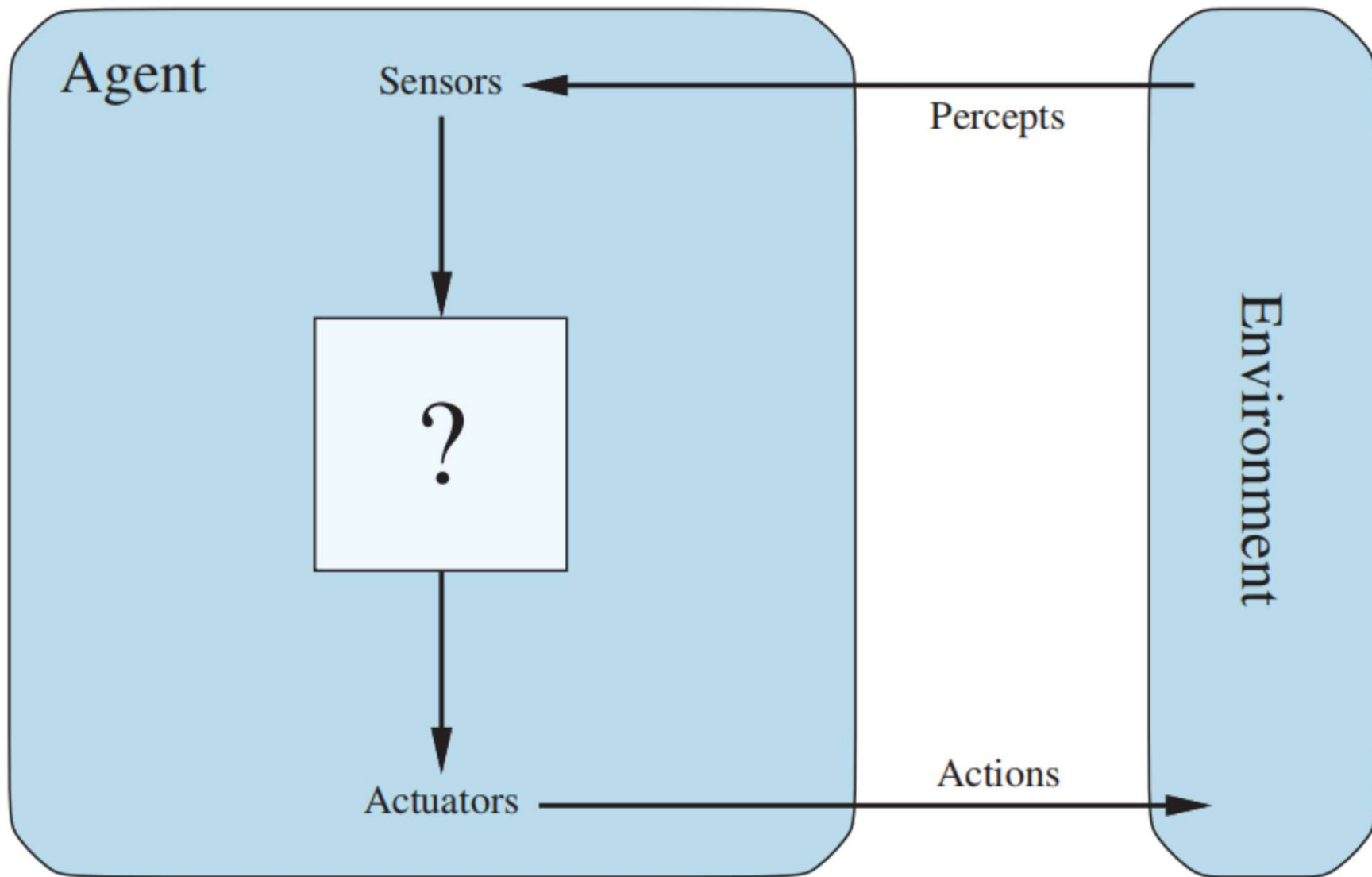
* no worries, we will make it a little less vague soon

AI: Constructing Agents

You can say that:

AI is focused on the **study and construction of agents that do the right thing.**

Agent



Agent

Agent:

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

Percepts and Percept Sequences

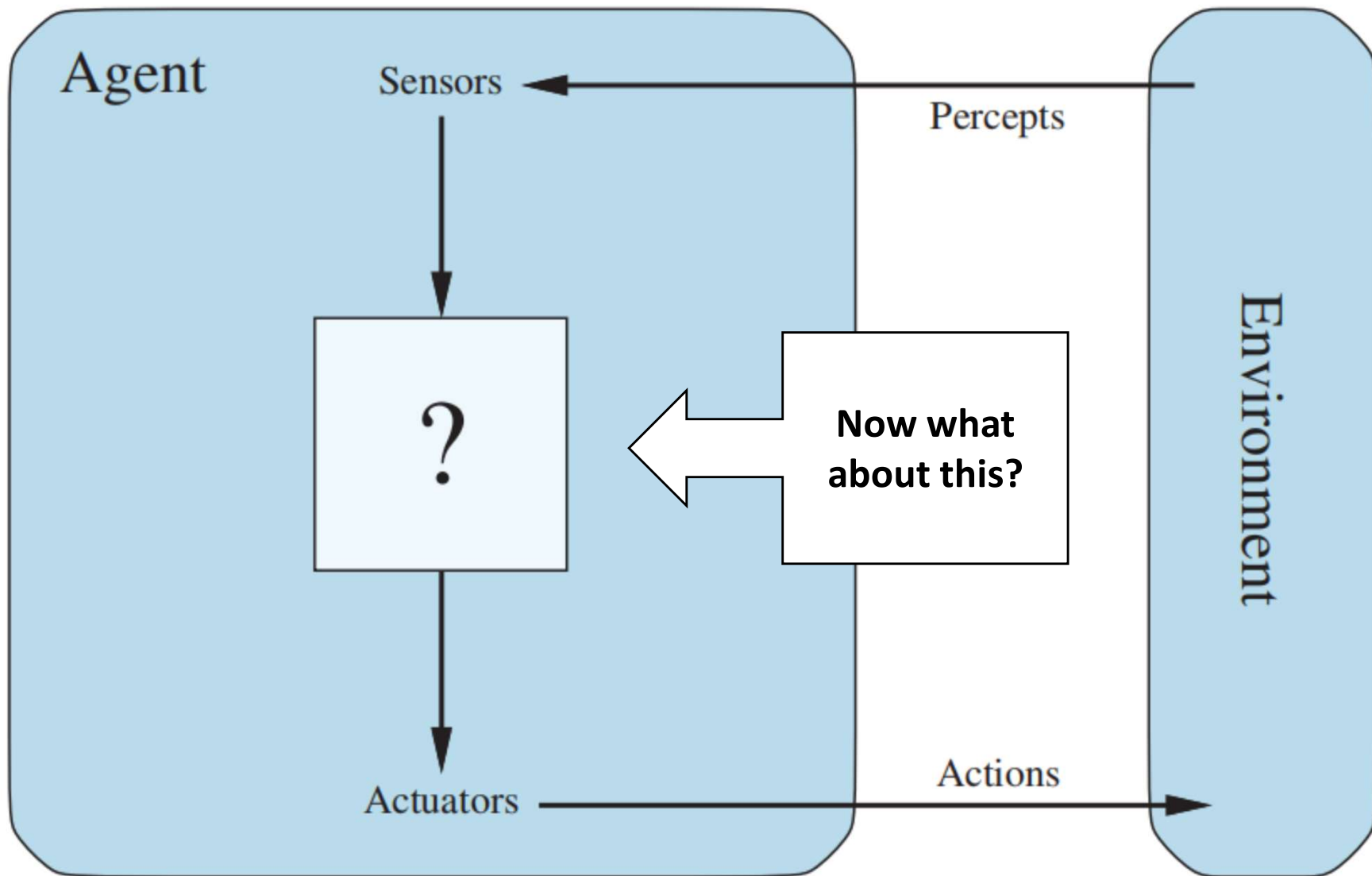
- **Percept:** content / information that agent's sensors are perceiving / capturing **currently**
- **Percept Sequence:** a **complete history** of **everything that agent has ever perceived**
 - any practical issues that you can see here?
 - what can a percept sequence be used for?

Percepts, Knowledge, Actions, States

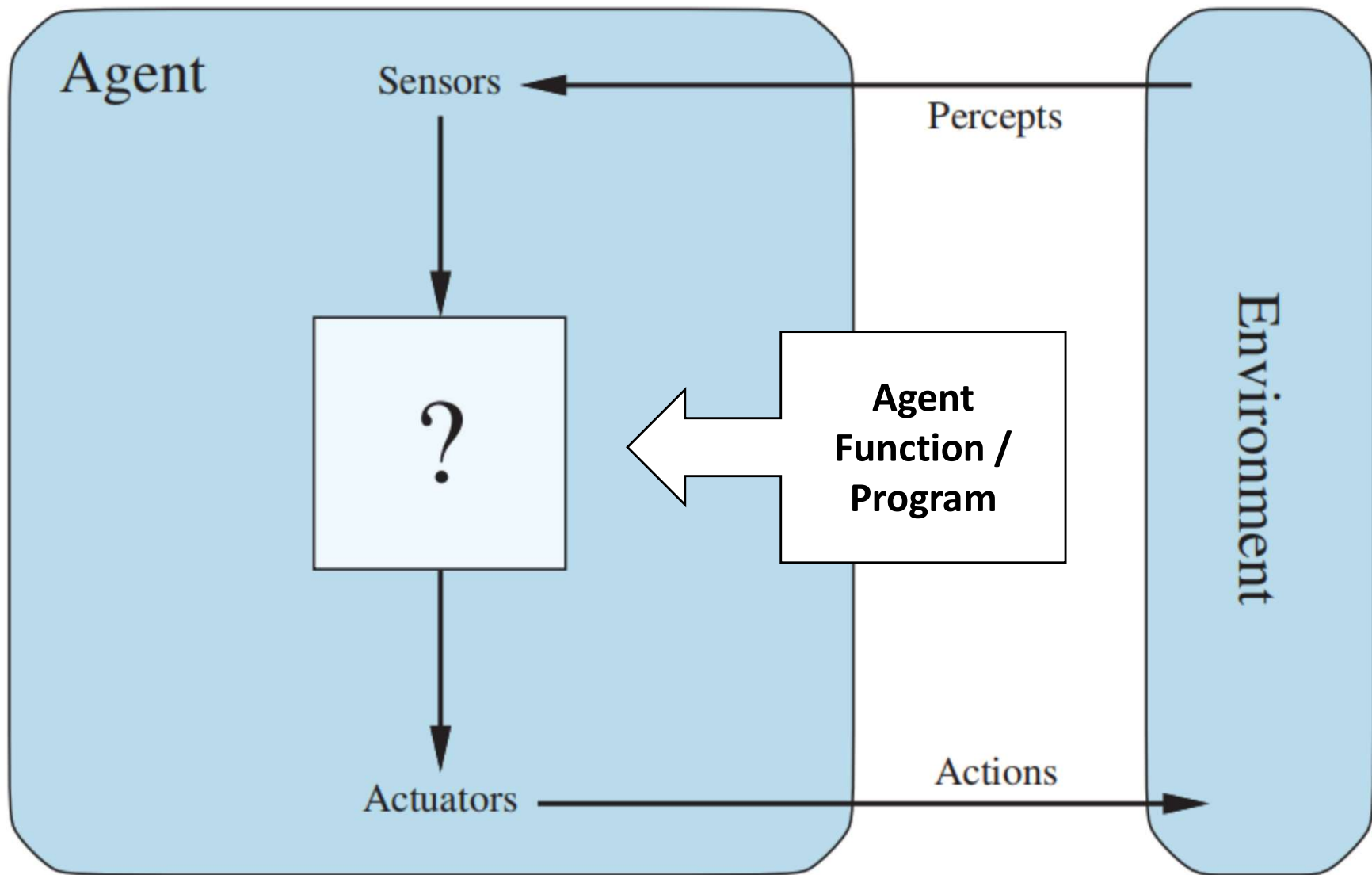
- Agent's choice of action / decision at any given moment:
 - CAN depend on:
 - built-in **knowledge**
 - entire **percept sequence**
 - CANNOT depend anything it hasn't perceived
- Agent's action CAN change the **environment state**

Knowledge is power, right?

Agent



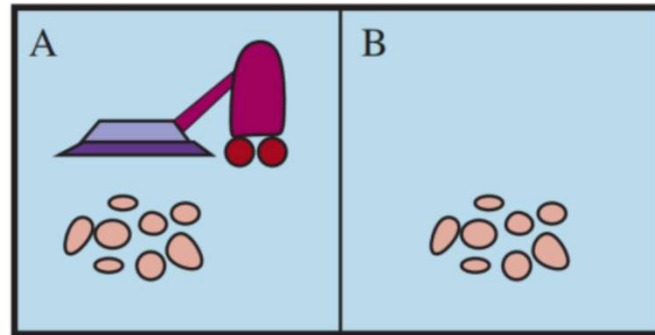
Agent Function / Program



Agent Function / Program

- Specifying an action choice for every possible percept sequence would define an agent
- Action \leftrightarrow percept sequence **mapping** IS the agent **function**
- Agent **function** describes agent **behavior**
- Agent **function** is an **abstract concept**
- Agent **program** implements agent **function**

Vacuum Cleaner Agent Example



Percept sequence	Action
$[A, \textit{Clean}]$	<i>Right</i>
$[A, \textit{Dirty}]$	<i>Suck</i>
$[B, \textit{Clean}]$	<i>Left</i>
$[B, \textit{Dirty}]$	<i>Suck</i>
$[A, \textit{Clean}], [A, \textit{Clean}]$	<i>Right</i>
$[A, \textit{Clean}], [A, \textit{Dirty}]$	<i>Suck</i>
\vdots	\vdots
$[A, \textit{Clean}], [A, \textit{Clean}], [A, \textit{Clean}]$	<i>Right</i>
$[A, \textit{Clean}], [A, \textit{Clean}], [A, \textit{Dirty}]$	<i>Suck</i>
\vdots	\vdots

Vacuum Cleaner Agent Example

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 \leftarrow LOOKUP(*percepts*, *table*)

return *action*

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*

Actions Have Consequences

- An agent can act upon its environment, but **how do we know if the end result is “right”?**
- After all, **actions have consequences**: either good or bad.
- Recall that **agent actions change environment state!**
- If state changes are desirable, an agent performs well.
- Performance measure evaluates state changes.

Performance Measure: A Tip

It is better to **design performance measures according to what one actually wants to be achieved in the environment**, rather than according to how one thinks the agent should behave.

Performance Measure: A Warning

If it is difficult to specify the performance measure, agents may end up optimizing a wrong objective. Handle uncertainty well in such cases.

Rationality

Rational decisions at the moment depend on:

- The **performance measure** that defines success criteria
- The agent's **prior knowledge** of the environment
- The **actions** that the agent can perform
- The agent's **percept sequence** so far

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 in Reality

- An omniscient agent will **ALWAYS** know the final outcome of its action. Impossible in reality. That would be perfection.
- Rationality maximizes what is **EXPECTED** to happen
- Perfection maximizes what **WILL** happen
- Performance can be improved by **information gathering and learning**

Designing the Agent for the Task

**Analyze the
Problem / Task
(PEAS)**

**Select Agent
Architecture**

**Select Internal
Representations**

**Apply
Corresponding
Algorithms**

Task Environment | PEAS

In order to start the agent design process we need to specify / define:

- The **P**erformance measure
- The **E**nvironment in which the agent will operate
- The **A**ctuators that the agent will use to affect the environment
- The **S**ensors that the agent will use to perceive the environment

PEAS: Taxi Driver Example

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: Other Examples

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

Task Environment Properties

Key dimensions by which task environments can be categorized:

- Fully vs partially observable (can be unobservable too)
- Single agent vs multiagent
 - multiagent: competitive vs. cooperative
- Deterministic vs. nondeterministic (stochastic)
- Episodic vs. sequential
- Static vs. dynamic
- Discrete vs. continuous
- Known vs. unknown (to the agent)

Fully Observable Environment



Source: Pixabay (www.pexels.com)

Freeciv.net

Map Nation Research Players Options Manual

Turn Done (59%)

Messages

Loulouse is finished building Archers.
 Year: 1/68 AD
 Archers lost to an attack by the Iranian Destroyer.
 Archers lost to an attack by the Iranian Destroyer.

Overview map

French Population: 440000 Turn: 259 Gold: 2183 (+74) Tax: 80 Lux: 20 Sci: 0

Illinois Institute of Technology

Partially Observable Environment



Source: https://en.wikipedia.org/wiki/Fog_of_war

Single-agent System



Source: cottonbro (www.pexels.com)

Single-agent System



Source: cottonbro (www.pexels.com)

Multiagent System



Source: Vlada Karpovich (www.pexels.com)

Multiagent System



Source: Vlada Karpovich (www.pexels.com)

Deterministic vs. Nondeterministic

- **Deterministic environment:**
 - next state is **completely determined** by the current state and agent action
 - deterministic AND fully observable environment: no need to worry about uncertainty
 - deterministic AND partially observable ***may*** appear nondeterministic
- **Nondeterministic (stochastic) environment:**
 - next state is **NOT completely determined** by the current state and agent action

Episodic vs. Sequential

- **Episodic environment:**
 - agent experience is divided into individual, **independent**, and atomic episodes
 - one percept - one action.
 - next action is not a function of previous action: not necessary to memorize it
- **Sequential environment:**
 - current decision / action **COULD** affect all future decisions / actions
 - better keep track of it

Static vs. Dynamic

- **Static environment:**
 - environment **CANNOT** change while the agent is taking its time to decide
- **Dynamic environment:**
 - environment **CAN** change while the agent is taking its time to decide -> decision / action may be dated
 - speed is important

Discrete vs. Continuous

- **Discrete environment:**
 - state changes are discrete
 - time changes are discrete
 - percepts are discrete
- **Continuous environment:**
 - state changes are continuous (“fluid”)
 - time changes are continuous
 - percepts / actions can be continuous

Known vs. Unknown (to Agent)

- **Known environment:**
 - agent **knows all outcomes** to its actions (or their probabilities)
 - agent “knows how the environment works”
- **Unknown environment:**
 - agent “doesn’t know all the details about the inner workings of the environment”
 - **learning and exploration** can be necessary

Task Environment 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

Hardest Case / Problem

- **Partially observable (incomplete information, uncertainty)**
- **Multiagent (complex interactions)**
- **Nondeterministic (uncertainty)**
- **Sequential (planning usually necessary)**
- **Dynamic (changing environment, uncertainty)**
- **Continuous (infinite number of states)**
- **Unknown (agent needs to learn / explore, uncertainty)**

Designing the Agent for the Task

**Analyze the
Problem / Task
(PEAS)**

**Select Agent
Architecture**

**Select Internal
Representations**

**Apply
Corresponding
Algorithms**

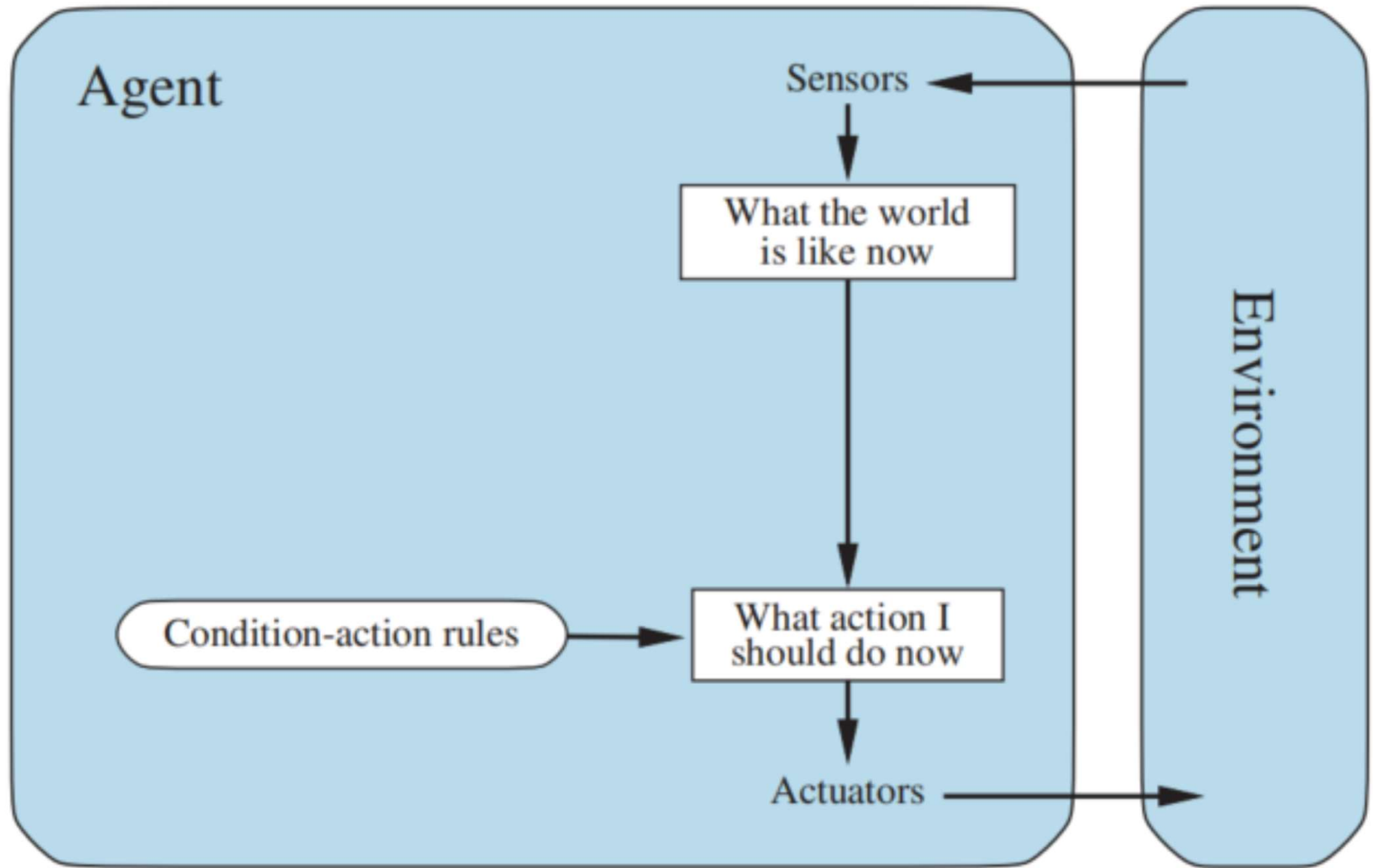
Agent Structure / Architecture

Agent = Architecture + Program

Typical Agent Architectures

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

Simple Reflex Agent

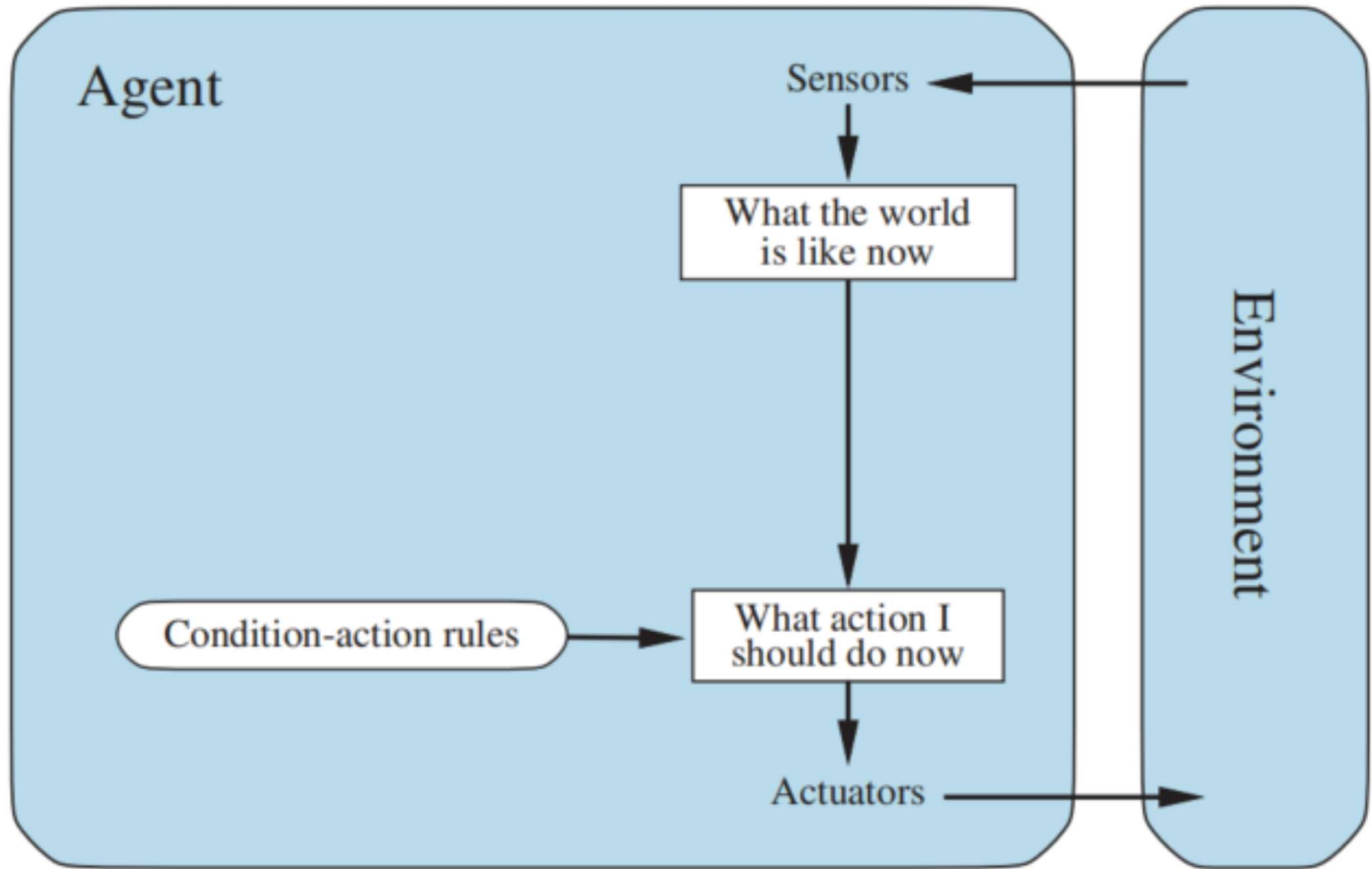


Simple Reflex Agent

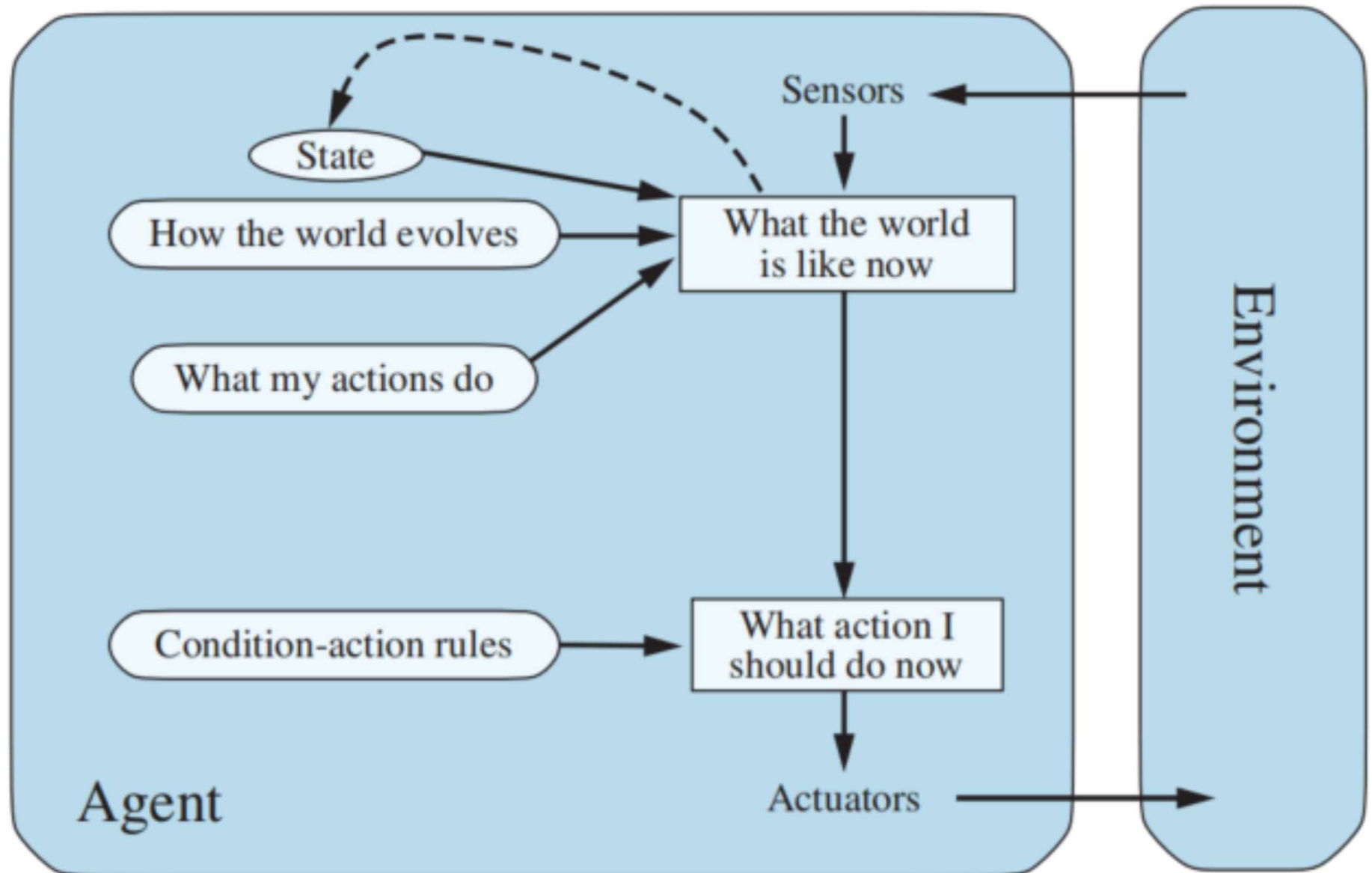
function SIMPLE-REFLEX-AGENT(*percept*) **returns** an action
persistent: *rules*, a set of condition–action rules

state \leftarrow INTERPRET-INPUT(*percept*)
rule \leftarrow RULE-MATCH(*state*, *rules*)
action \leftarrow *rule*.ACTION
return *action*

Simple Reflex Agent: Challenges?



Model-based Reflex Agent



Model-based Reflex Agent

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

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

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

sensor_model, a description of how 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

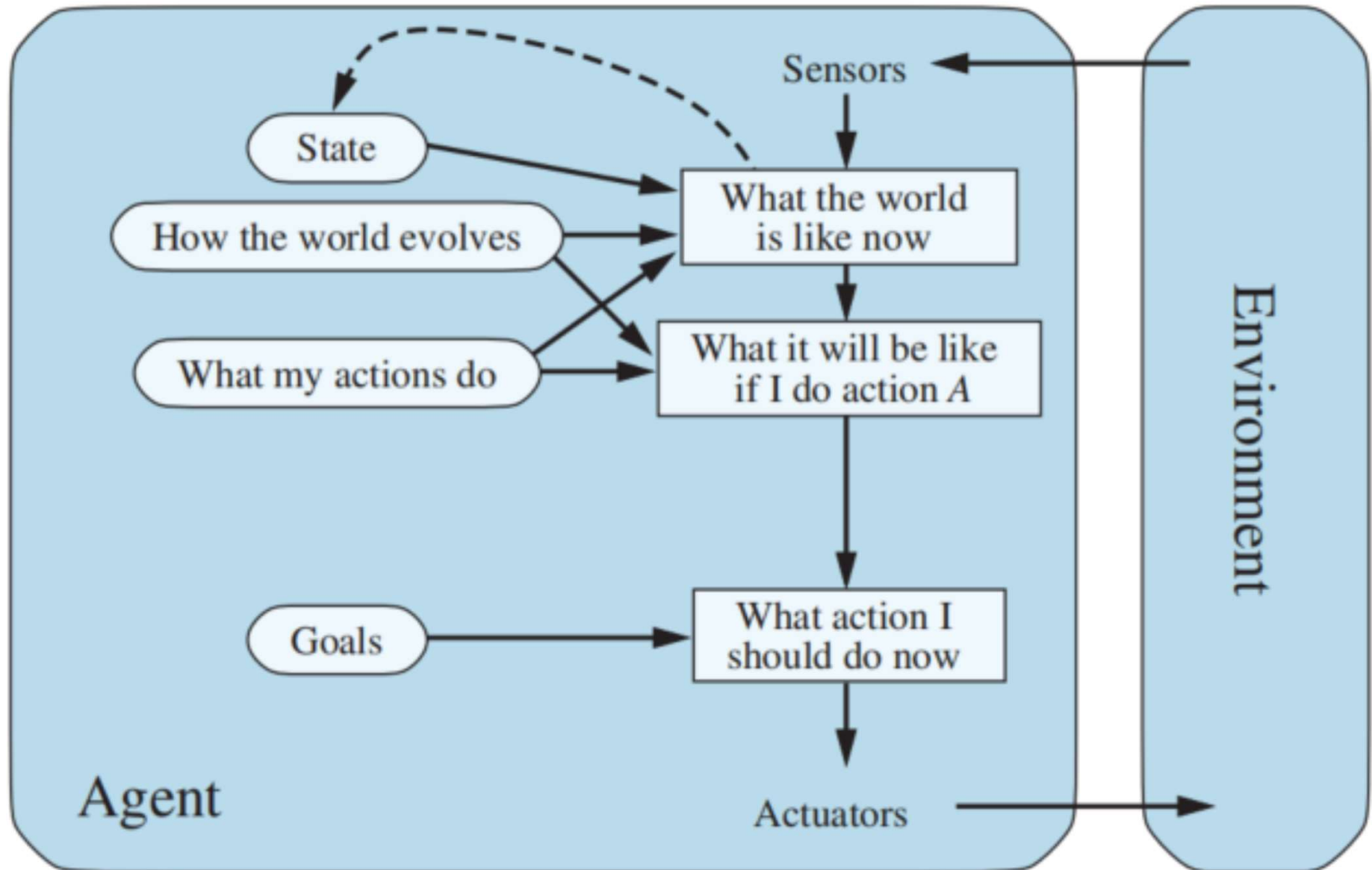
state \leftarrow UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)

rule \leftarrow RULE-MATCH(*state*, *rules*)

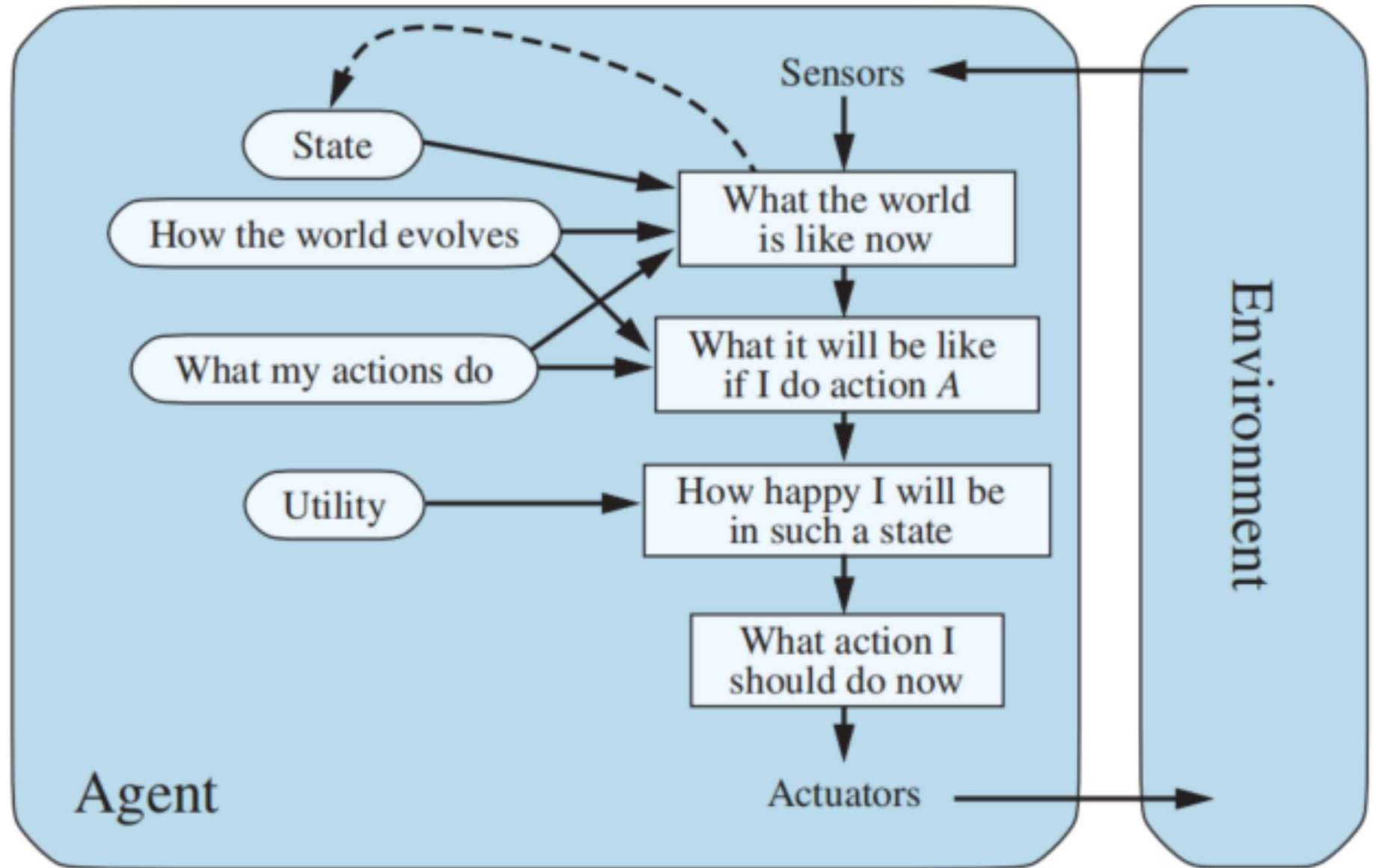
action \leftarrow *rule*.ACTION

return *action*

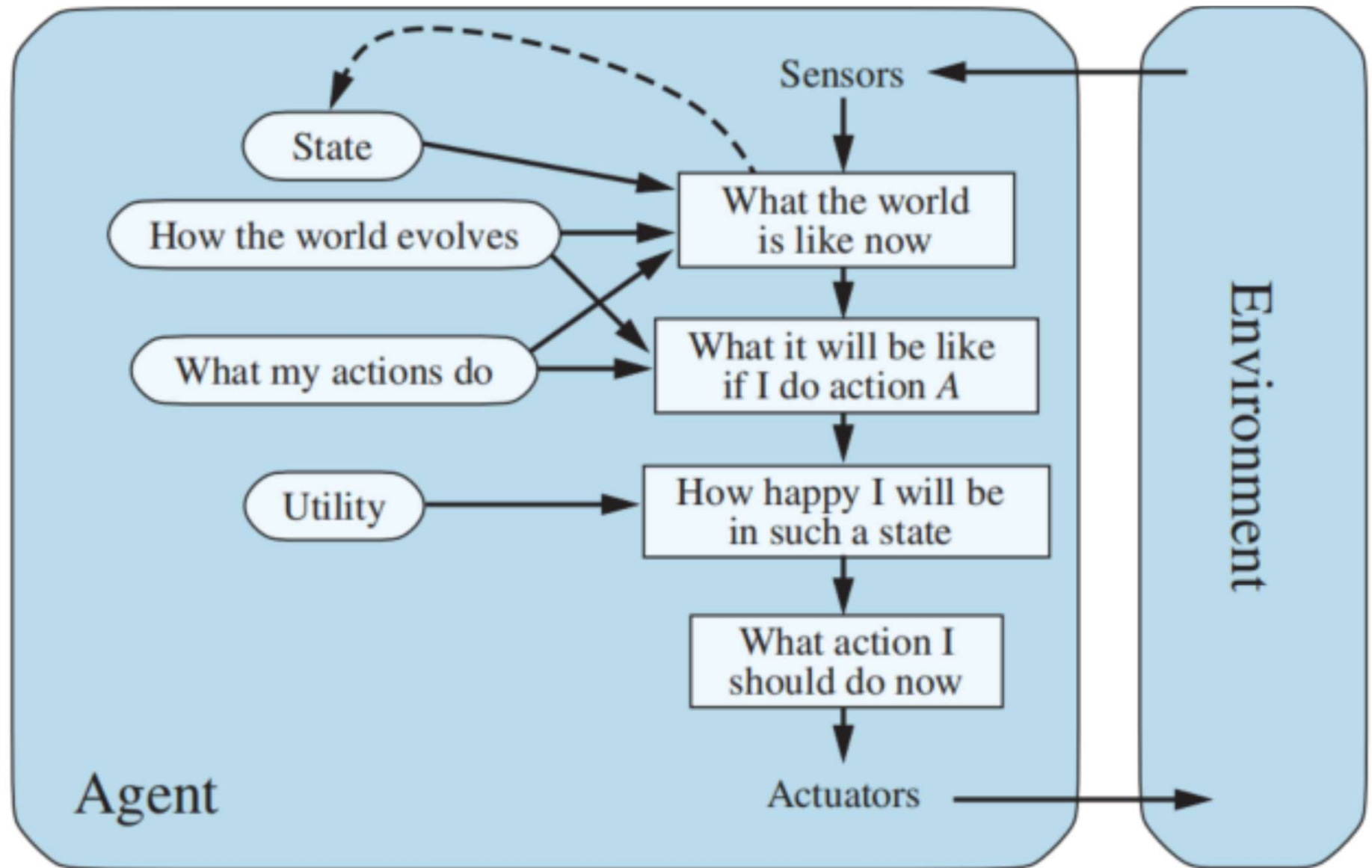
Model-based Goal-based Agent



Model-based Utility-based Agent



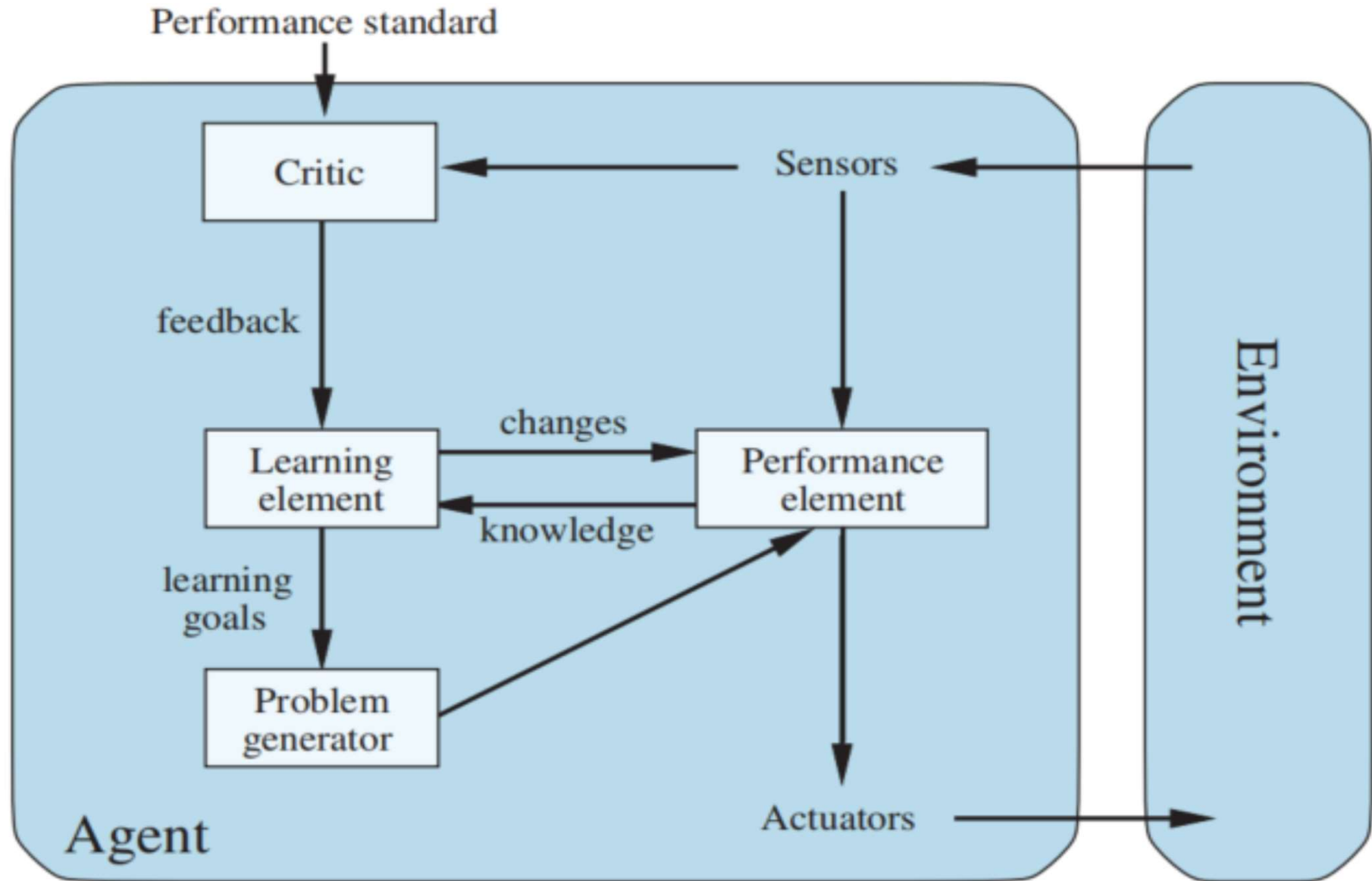
Model-based Agents: Challenges?



Typical Agent Architectures

- **Simple reflex agent:** uses condition-action rules
- **Model-based reflex agent:** keeps track of the unobserved parts of the environment by maintaining internal state:
 - “how the world works”: state transition model
 - how percepts and environment is related: sensor model
- **Goal-based reflex agent:** maintains the model of the world and goals to select decisions (that lead to goal)
- **Utility-based reflex agent:** maintains the model of the world and utility function to select **PREFERRED** decisions (that lead to the best expected utility: $\text{avg} (EU * p)$)

Learning Agent



Designing the Agent for the Task

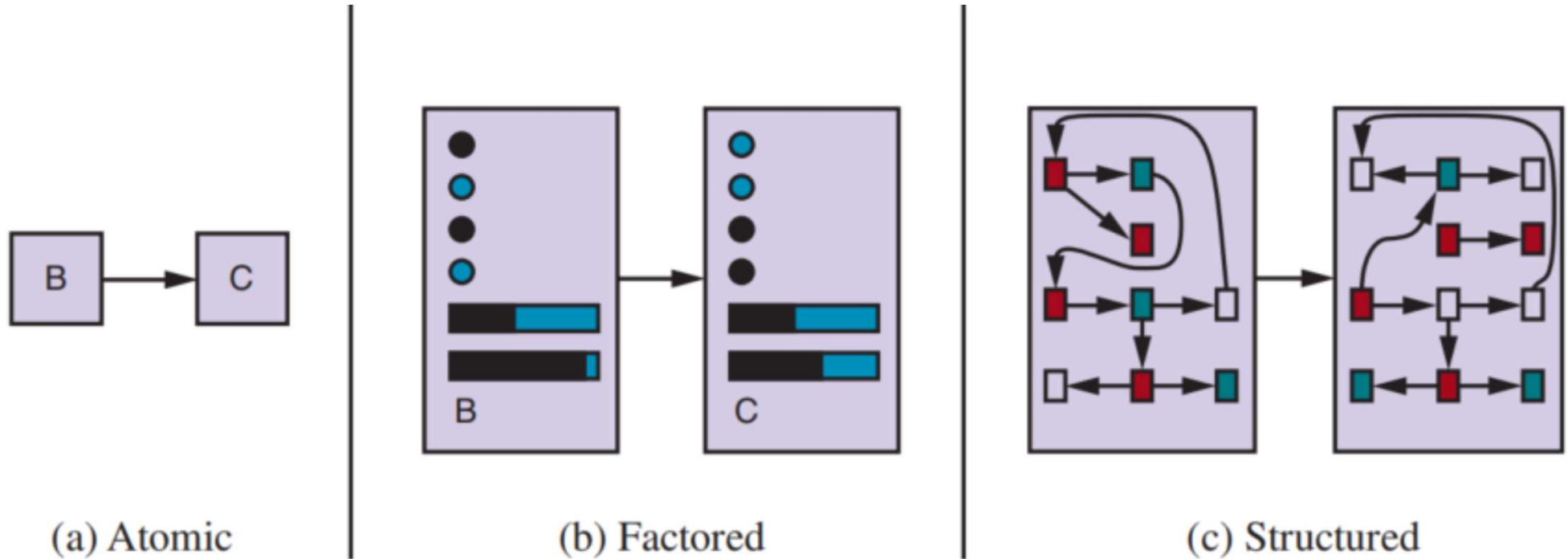
**Analyze the
Problem / Task
(PEAS)**

**Select Agent
Architecture**

**Select Internal
Representations**

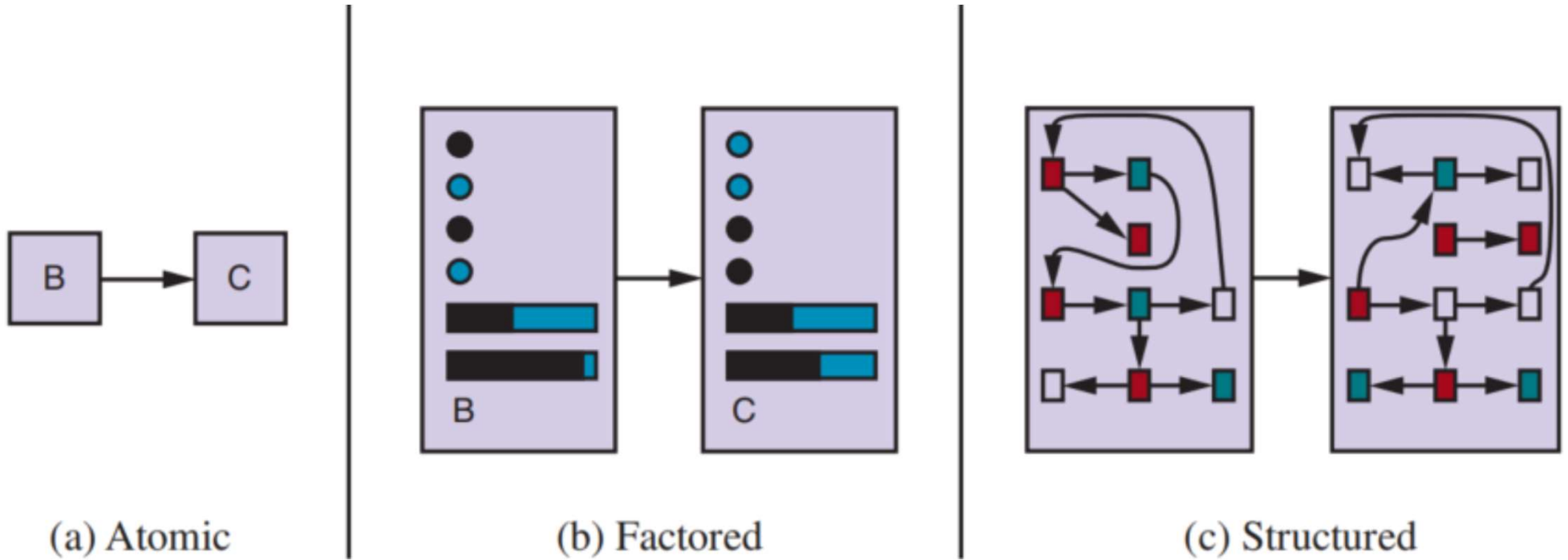
**Apply
Corresponding
Algorithms**

State and Transition Representations



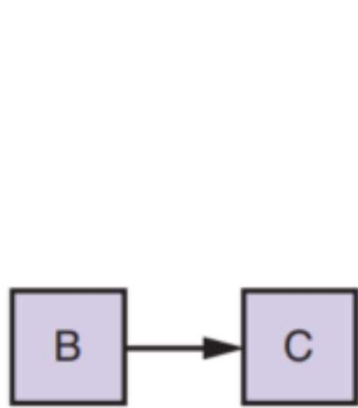
- **Atomic:** state representation has NO internal structure
- **Factored:** state representation includes fixed attributes (which can have values)
- **Structured:** state representation includes objects and their relationships

State and Transition Representations

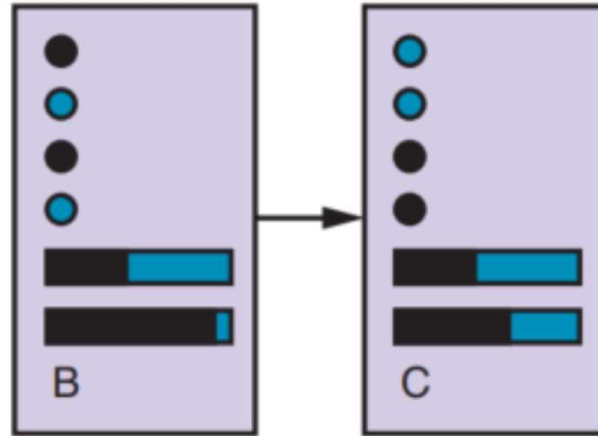


Complexity, level of detail, expresiveness, more difficult to process

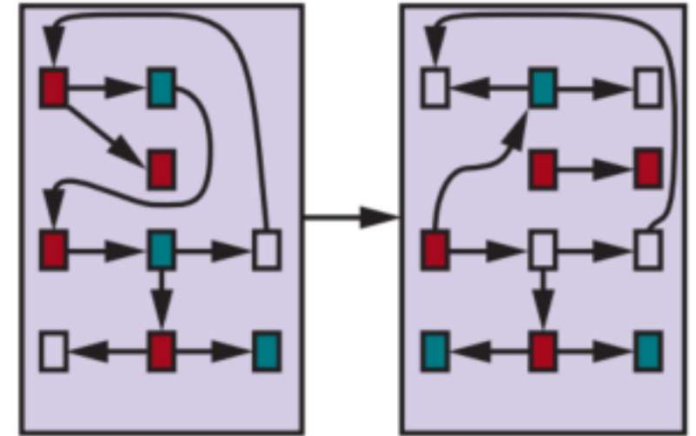
Representations and Algorithms



(a) Atomic



(b) Factored



(c) Structured

- Searching
- Hidden Markov models
- Markov decision process
- Finite state machines

- Constraint satisfaction algorithms
- Propositional logic
- Planning
- Bayesian algorithms
- Some machine learning algorithms

- Relational database algorithms
- First-order logic
- First-order probability models
- Natural language understanding (some)

Designing the Agent for the Task

**Analyze the
Problem / Task
(PEAS)**

**Select Agent
Architecture**

**Select Internal
Representations**

**Apply
Corresponding
Algorithms**