CHAPTER 2

Intelligent Agents

Outline

- Definition of an agent.
- Task environment.
- Environment types.
- Agent types.

Book Code

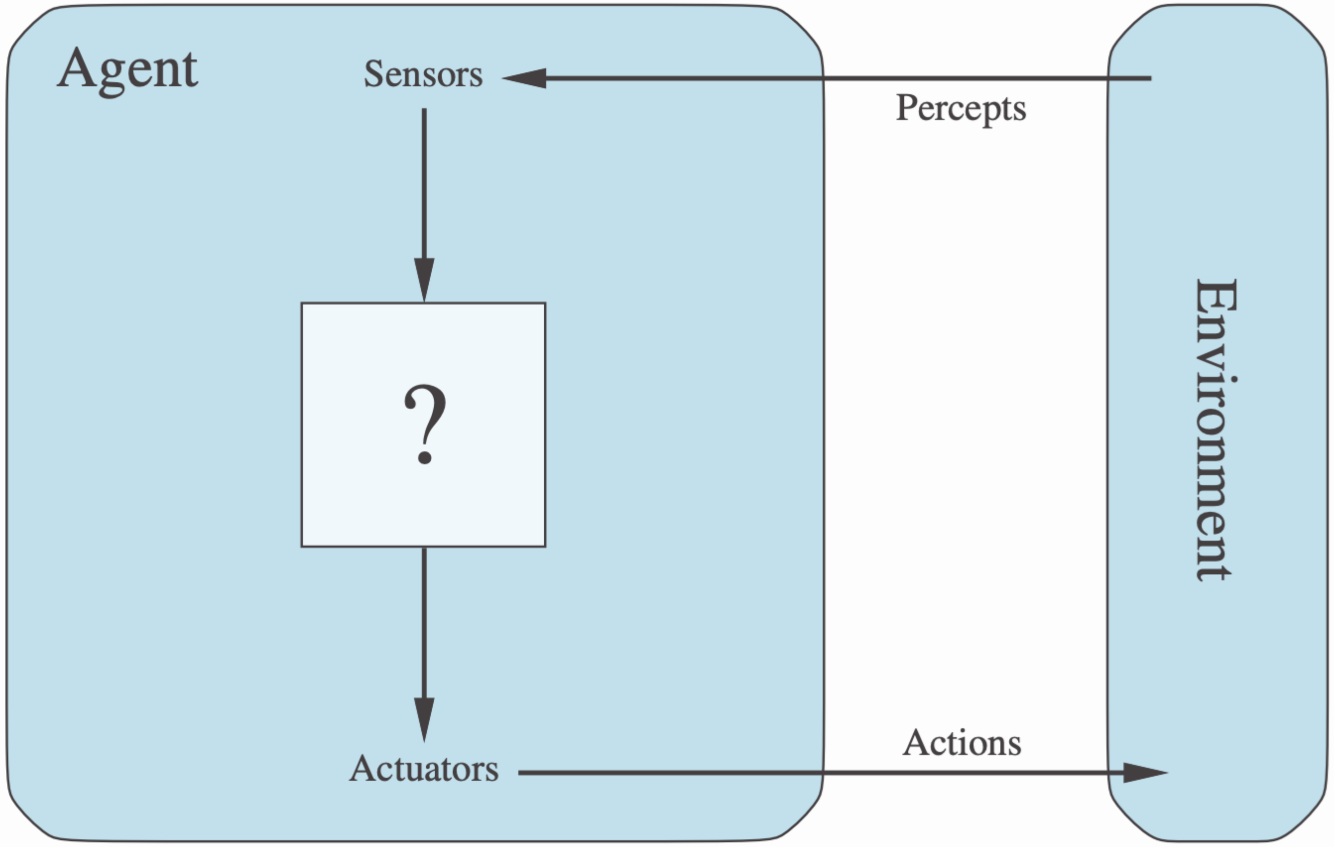
- https://github.com/aimacode

- https://mybinder.org/repo/aimacode/aima-python

What is an Agent?

- We have seen that our goal in AI is to create intelligent (rational) agents.
- But what is an agent?
  - An agent is anything that perceives its environment through sensors and acts upon it through actuators

What is an Agent?



Percept: Percept refers to the content an agent’s sensors are perceiving.

Actions: 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.

Sensors: The environmen of the system is observed by intelligent agents through sensors.

Actuators: These are components throught which energy is converted to motion. They perform the role of controlling and moving a system.

Environment: Environment is the surrounding of the agent.

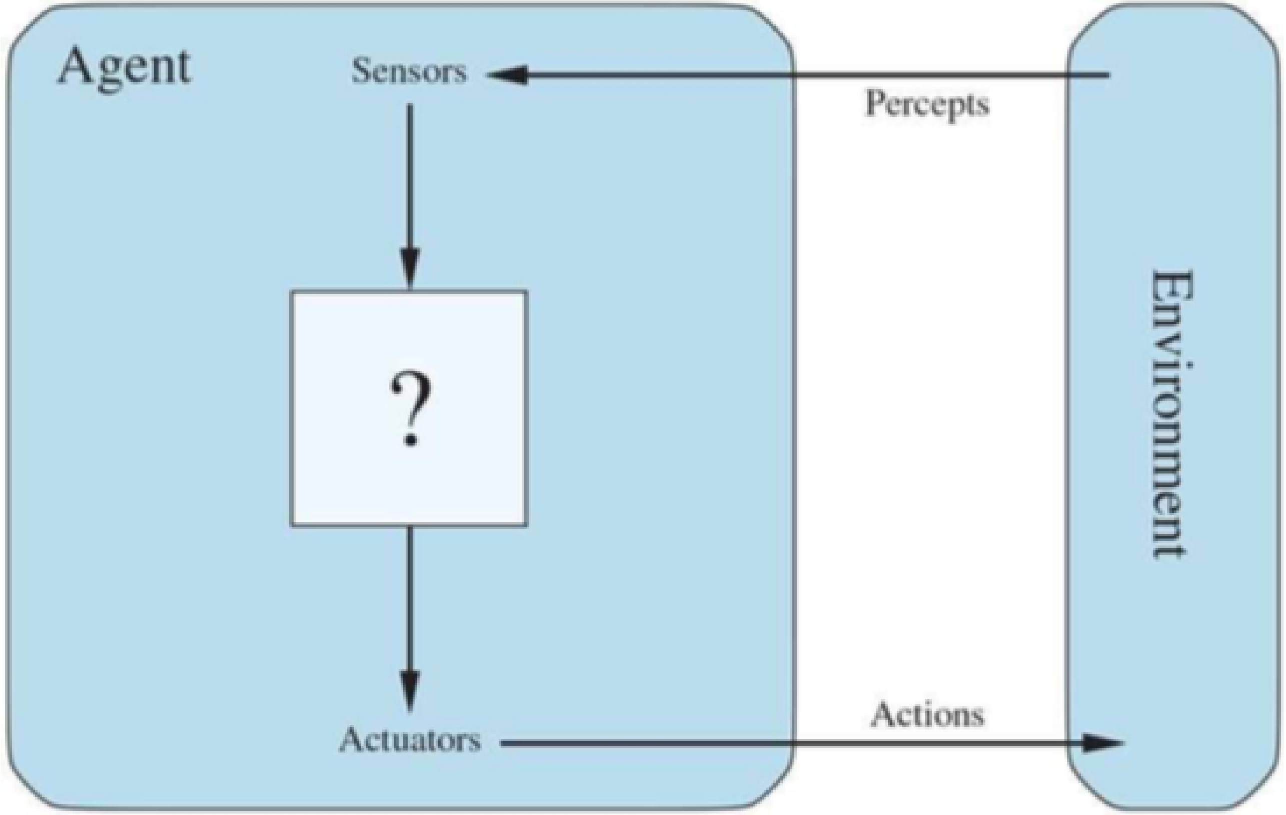
Agents and Environments

Human agent: A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.

Robotic agent: A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.

Software agent: A software agent receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, sending network packets, and displaying information or generating sounds.

Environment: The environment could be everything—the entire universe! In practice it is just that **part of the universe whose state we care about** when designing this agent—the part that affects what the agent perceives and that is affected by the agent's actions.



Agents interact with environments through sensors and actuators.

How does an agent work?

- **Percept sequence**: all the history of inputs to the agent.
- The actions of an agent may be a function of all this history.
- The agent can be seen as having a **function** that maps inputs to outputs: precept sequence to actions.
- The function is internally implemented as a **program**.

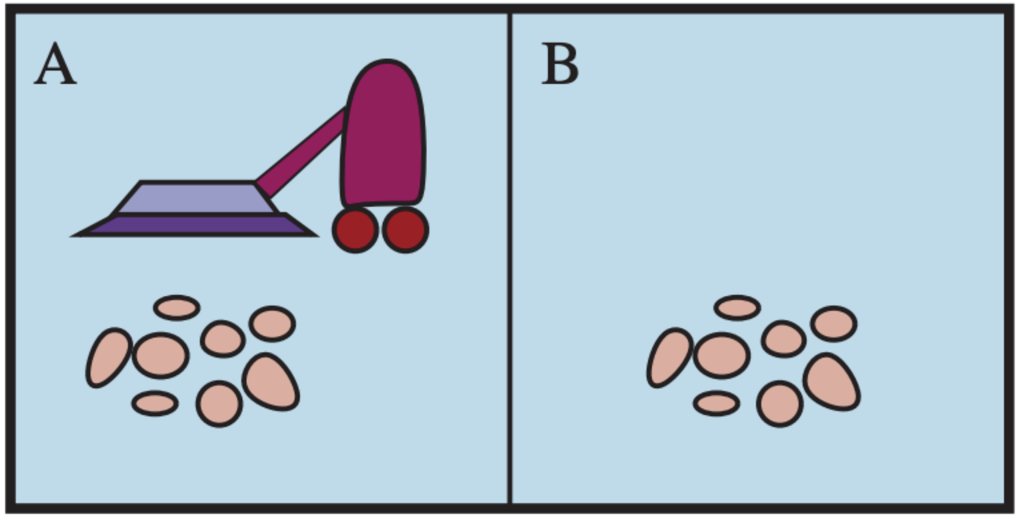
Example: the vacuum world

The vacuum-cleaner world, which consists of a robotic vacuum-cleaning agent in a world consisting of squares that can be either dirty or clean.

Figure shows a configuration with just two squares, A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square.

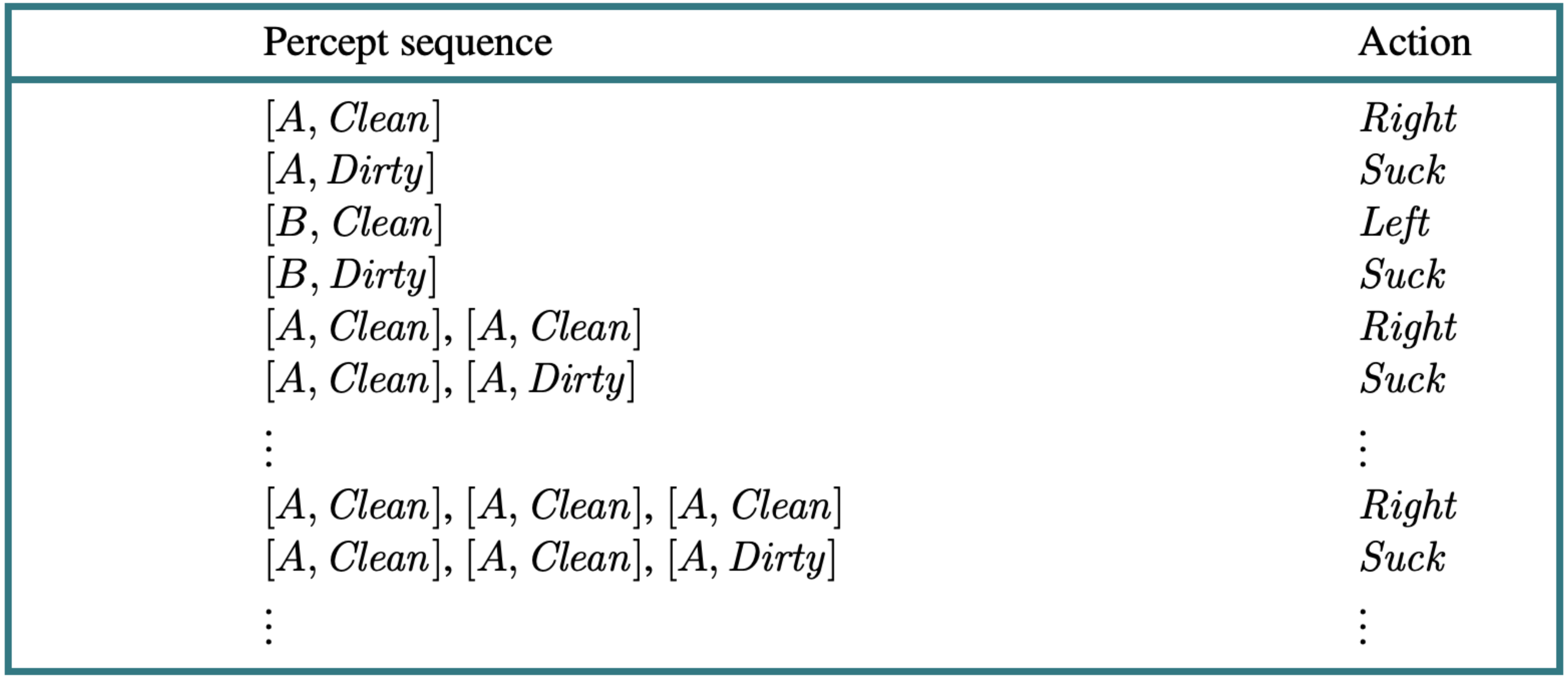
Steps:

- The agent starts in square A.
- The available actions are to move to the right, move to the left, suck up the dirt, or do nothing.
- One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square.



- Percepts: location and contents, e.g., [A;Dirty]
- Actions: Left, Right, Suck, NoOp

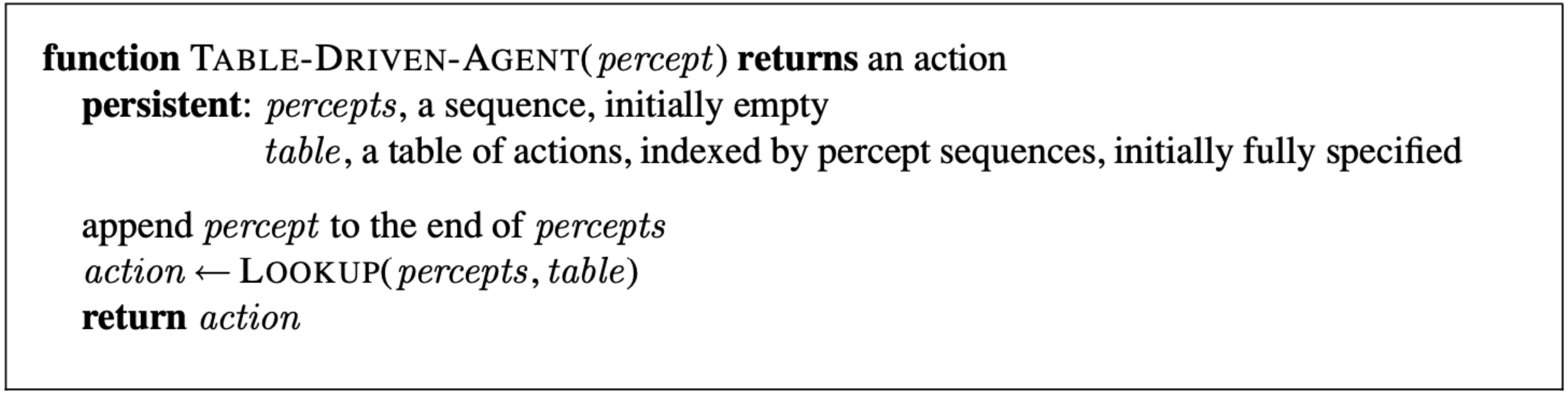
The vacuum-cleaner agent



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*
⋮	⋮

- We can define different agents by changing the actions on the second column → What is the best agent?
- https://mybinder.org/repo/aimacode/aima-python

Table-Driven Vacuum Cleaner Agent



```
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
```

- Must construct a table that contains the appropriate action for every possible percept sequence
- P: set of possible percepts, T: lifetime of the agent (the total number of percepts it will receive). The lookup table will contain:

$$\sum_{t=1}^{T} |P|^t \quad ; \quad P = \{[A, Clean], [A, Dirty], [B, Clean], [B, Dirty]\}$$

Rationality

- Fixed performance measure evaluates the environment sequence
  - One point per square cleaned up in time T?
    - A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again
  - One point per clean square per time step, minus one per move?

Good Behaviour: The concept of Rationality

A **rational agent** is one that does the right thing. "*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.* "

Evaluation of Good Behaviour:

1. Performance Measures
2. Rationality
3. Omniscience, learning, and autonomy

An **omniscient agent** knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality.

PEAS

- To design a rational agent, we need to define:
  - The **P**erformance measure.
  - The **E**nvironment.
  - The **A**ctuators.
  - The **S**ensors.
- These are called the **task environment**.

PEAS for an automated **taxi**

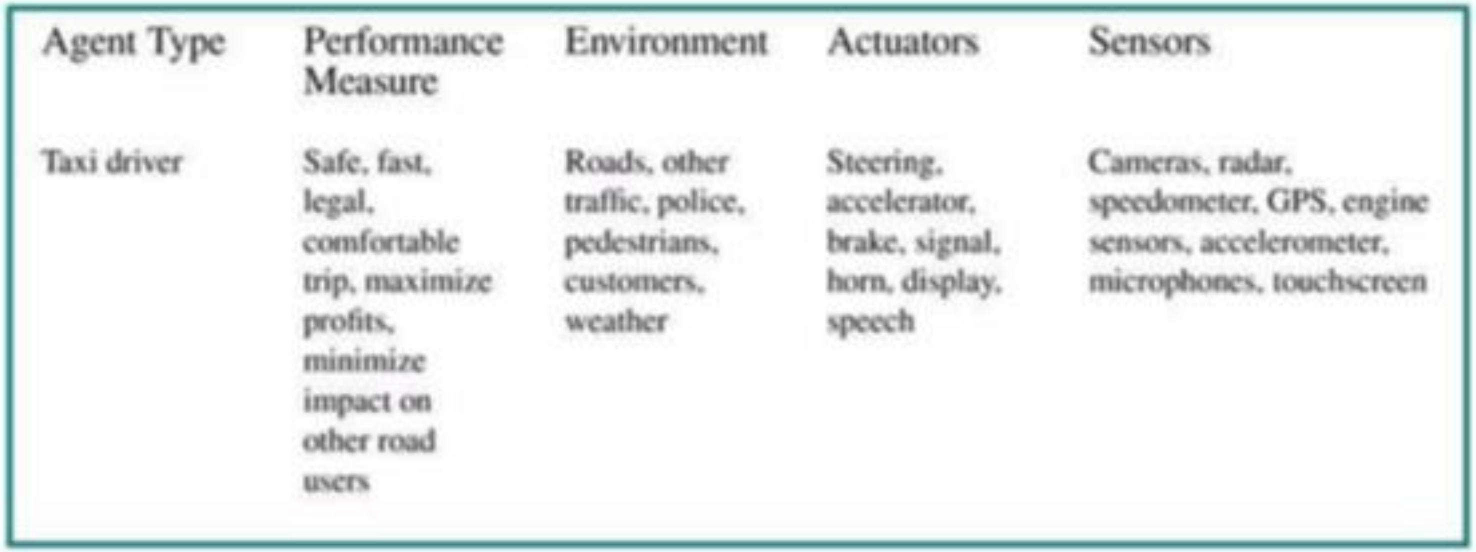
- Performance measure: safety, destination, profits, legality, comfort …
- Environment: streets/freeways, traffic, pedestrians, weather…
- Actuators: steering, accelerator, brake, horn, speaker/display…
- Sensors: video, accelerometers, gauges, engine sensors, keyboard, GPS…

The Nature Of Environment

PEAS (Performance, Environment, Actuators, Sensors)

PEAS Example: Automated taxi driver

- what is the performance measure to which we would like our automated driver to aspire?
- what is the driving environment that the taxi will face?
- The actuators for an automated taxi include those available to a human driver.
- The basic sensors for the taxi will include one or more video cameras so that it can see.



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 driver.

PEAS for an **Internet shopping agent**

- Performance measure: price, quality, appropriateness, efficiency
- Environment: current and future WWW sites, vendors, shippers
- Actuators: display to user, follow URL, fill in forms
- Sensors: HTML pages (text, graphics, scripts)

PEAS

- Agent: Medical diagnosis system
- Performance measure: Healthy patient, minimize costs, lawsuits
- Environment: Patient, hospital, staff
- Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
-
- Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS

- Agent: Part-picking robot
- Performance measure: Percentage of parts in correct bins
- Environment: Conveyor belt with parts, bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

PEAS

- Agent: Interactive English tutor
- Performance measure: Maximize student's score on test
- Environment: Set of students
- Actuators: Screen display (exercises, suggestions, corrections)
- Sensors: Keyboard

Environment types

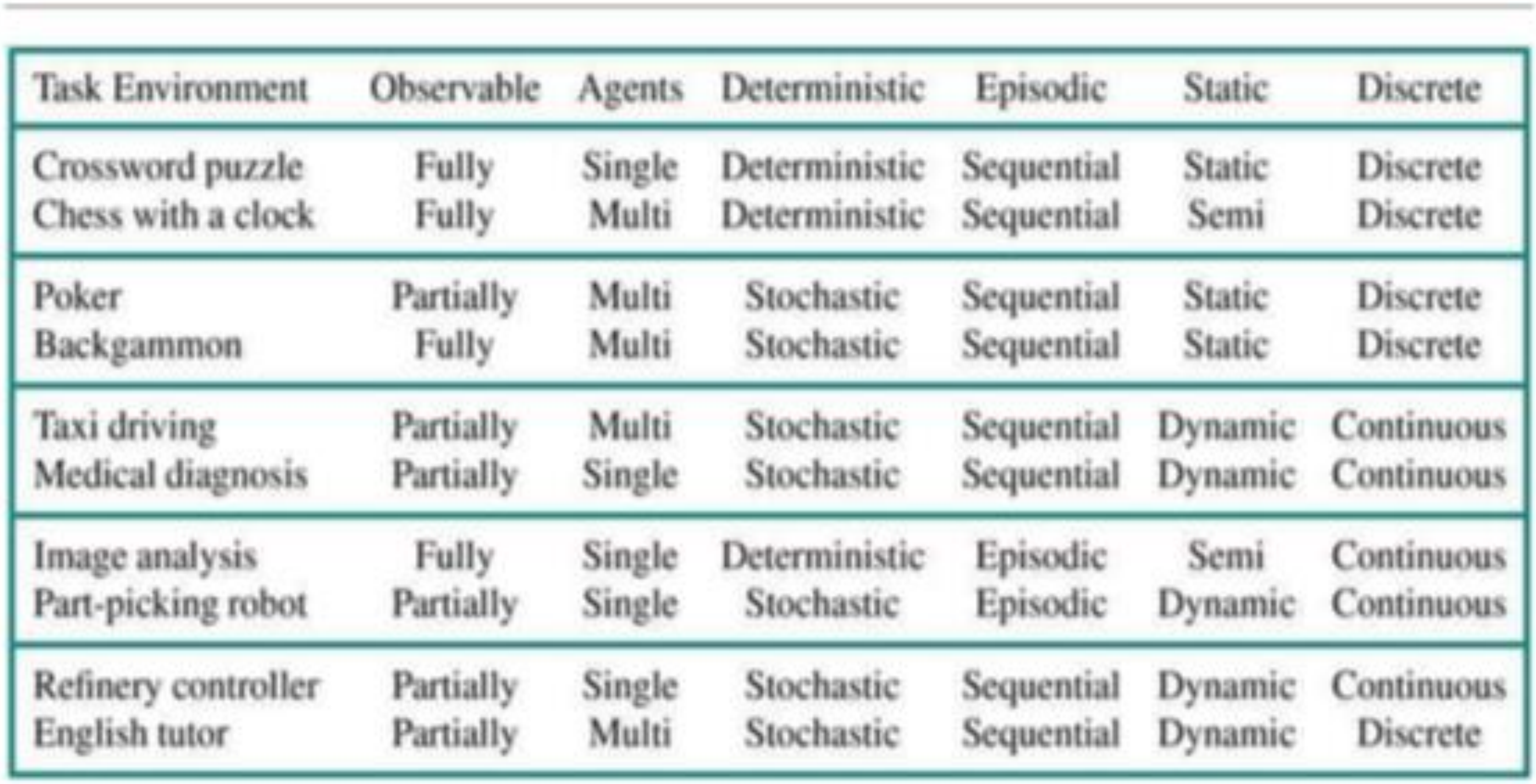
- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
-
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
-
- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multiagent): An agent operating by itself in an environment.

Properties/Types of environment

- FULLY OBSERVABLE VS. PARTIALLY OBSERVABLE
- SINGLE-AGENT VS. MULTIAGENT
- EPISODIC VS. SEQUENTIAL
- STATIC VS. DYNAMIC
- DISCRETE VS. CONTINUOUS
- KNOWN VS. UNKNOWN
- Deterministic vs Stochastic

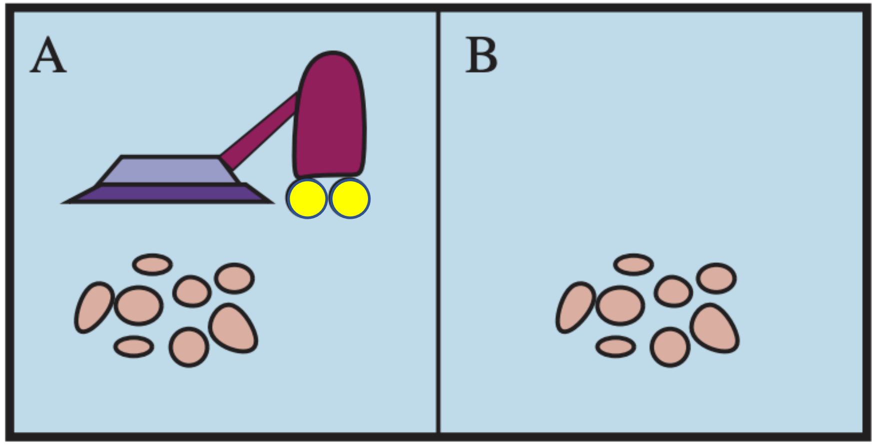


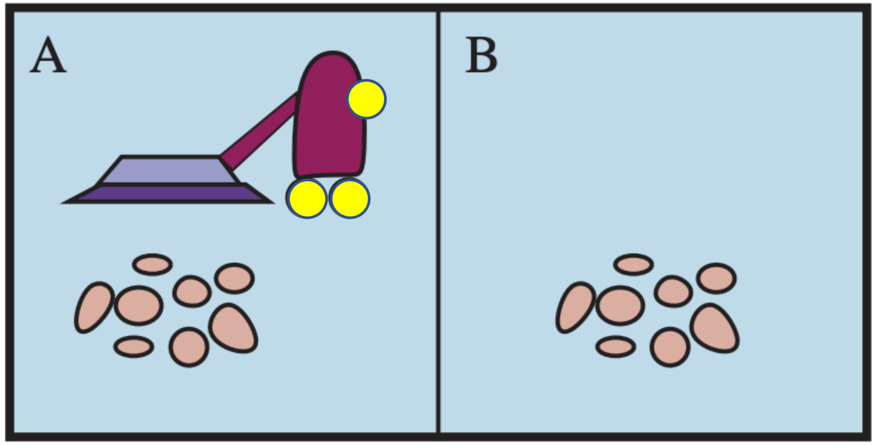
Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	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.

Properties of task environments

- **Fully observable**: if the agent’s sensor can observe the complete state of the environment. Otherwise **partially observable**.
  - Question: Can the agent sense everything (know all there is to know)?

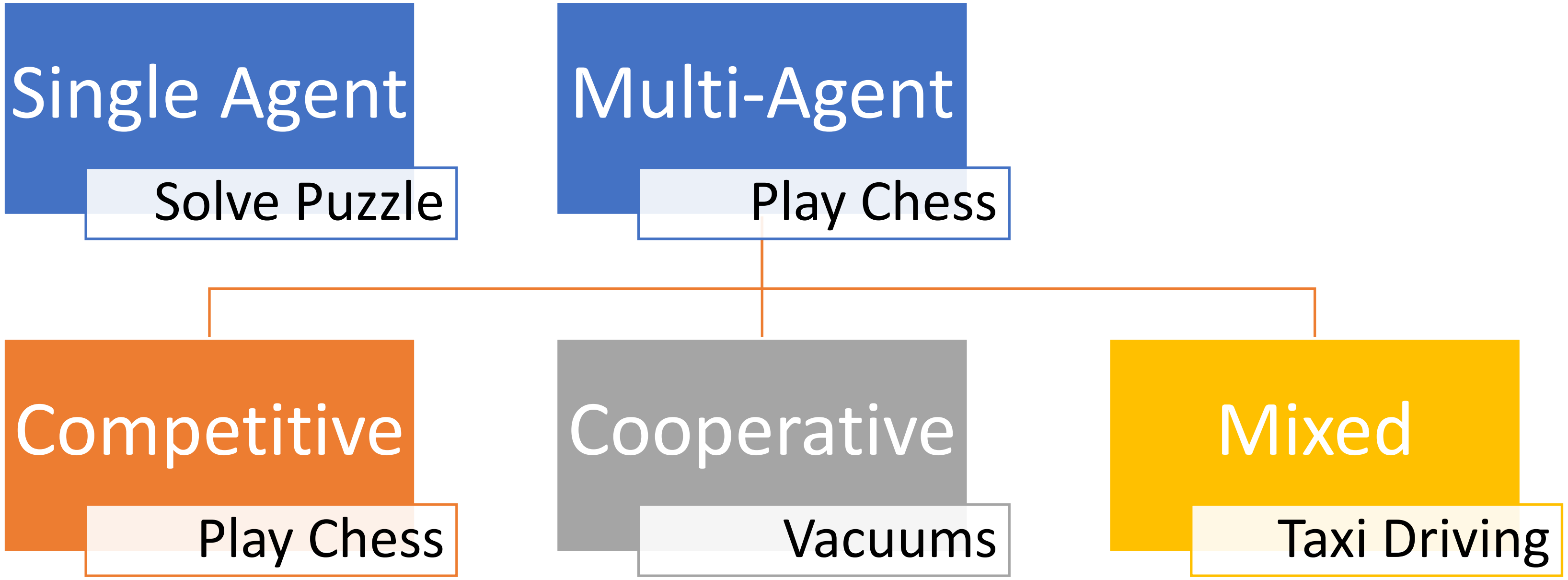




Sensors only underneath vs. sensors underneath and to the side

Properties of task environments

- **Single-agent** vs. **multi-agent**.
  - Question: is there only one agent or more than one?



Properties of task environments

- **Deterministic**: next state fully determined by the current state and the action. Otherwise **stochastic** **(chance or probability vs. deterministic with no probability)**.
  - Question: Does what happens next have any probability? Or do we know 100% what will happen?
- **Episodic**: atomic episodes, each episode does not depend on the past episodes (assembly line robot). Otherwise **sequential**.
  - Question: Does what happened before (the past history) affect what the agent will do?

Properties of task environments

- **Static**: changes only by the actions of the agent (Puzzle), otherwise **dynamic** (Taxi-driving**).** **Semi-dynamic** (Time based chess)
  - Question: Can the environment change while the agent is deciding what to do?
- **Discrete**: does not change continuously in time. A discrete environment is one where you have finitely many action choices, and finitely many things you can sense (Chess). Otherwise **continuous** (Taxi Driving)**.**
  - Question: Are the locations continuous or discrete?

Properties of task environments

- **Known** vs. **unknown**: this distinction refers not to the environment itself but to the agent’s (or designer’s) state of knowledge about the “laws of physics” of the environment.
  - Known partially observable: solitaire (know all rules, can’t see cards)
  - Unknown fully observable: a new video game (can see everything but don’t know buttons)

The Structure of Agents

- The job of AI is to design an **agent program** that implements the agent function: the mapping from percepts to actions.
- We assume this program will run on some sort of computing device with physical sensors and actuators: we call this the **architecture**:

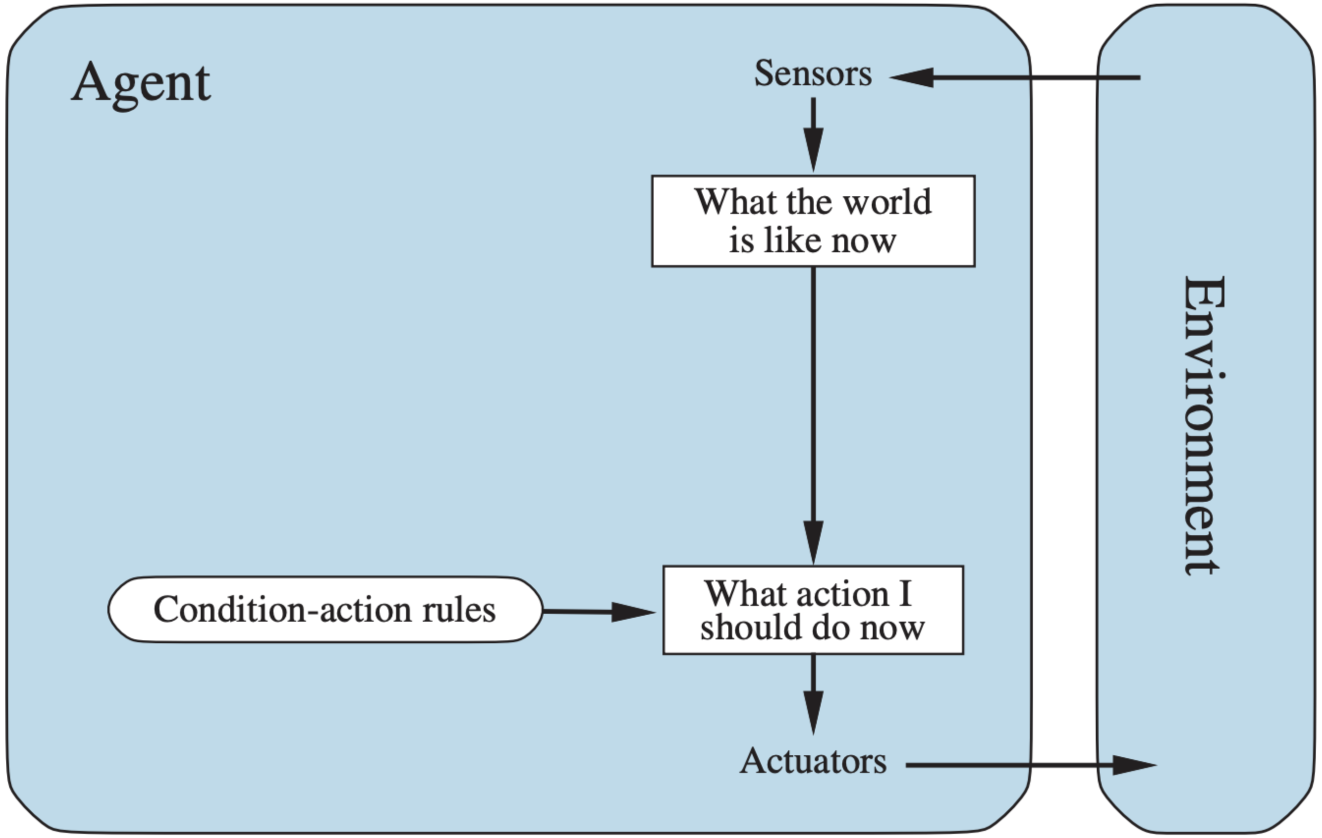
agent = architecture + program

Agent types

- **The environment type largely determines the agent design**

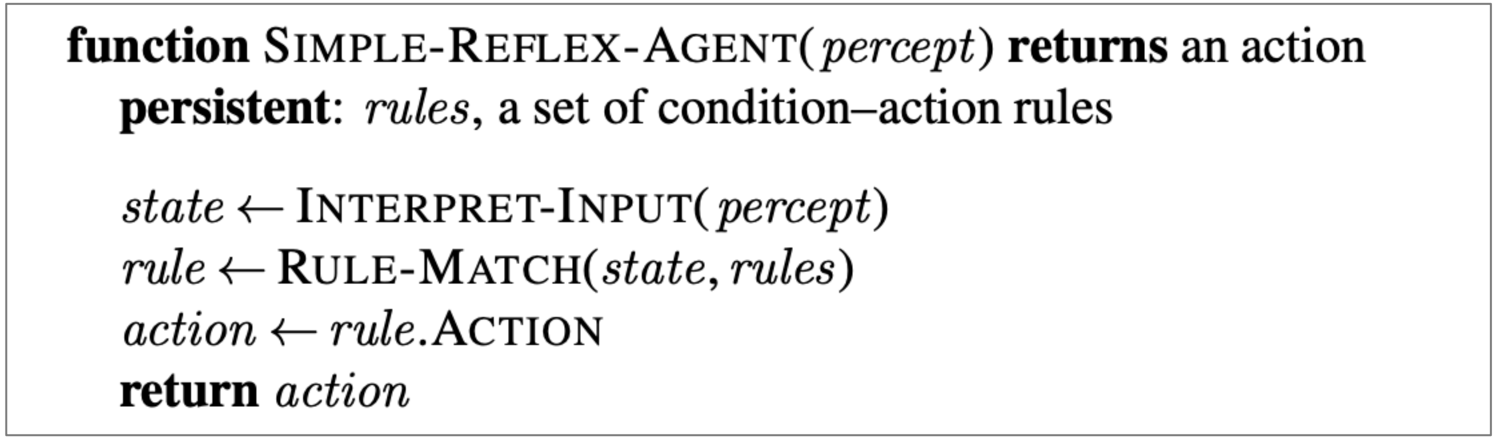
- Four basic types in order of increasing generality:
  1. Simple reflex agents
  2. Model-based reflex agents
  3. Goal-based agents
  4. Utility-based agents
- All these can be turned into
  5. Learning agents

1. Simple reflex agents



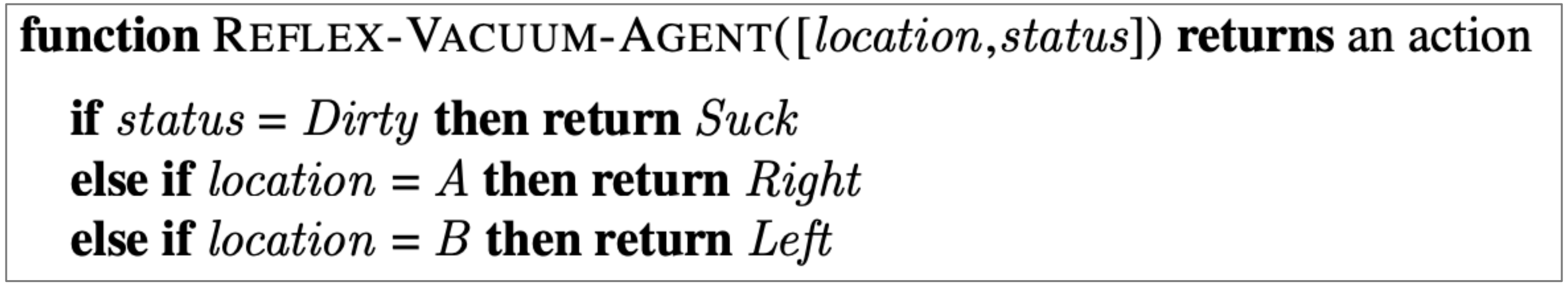
Select actions based on the *current* percept, ignoring the rest of the percept history.

Example of 1. Simple Reflex Agent



```
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
```

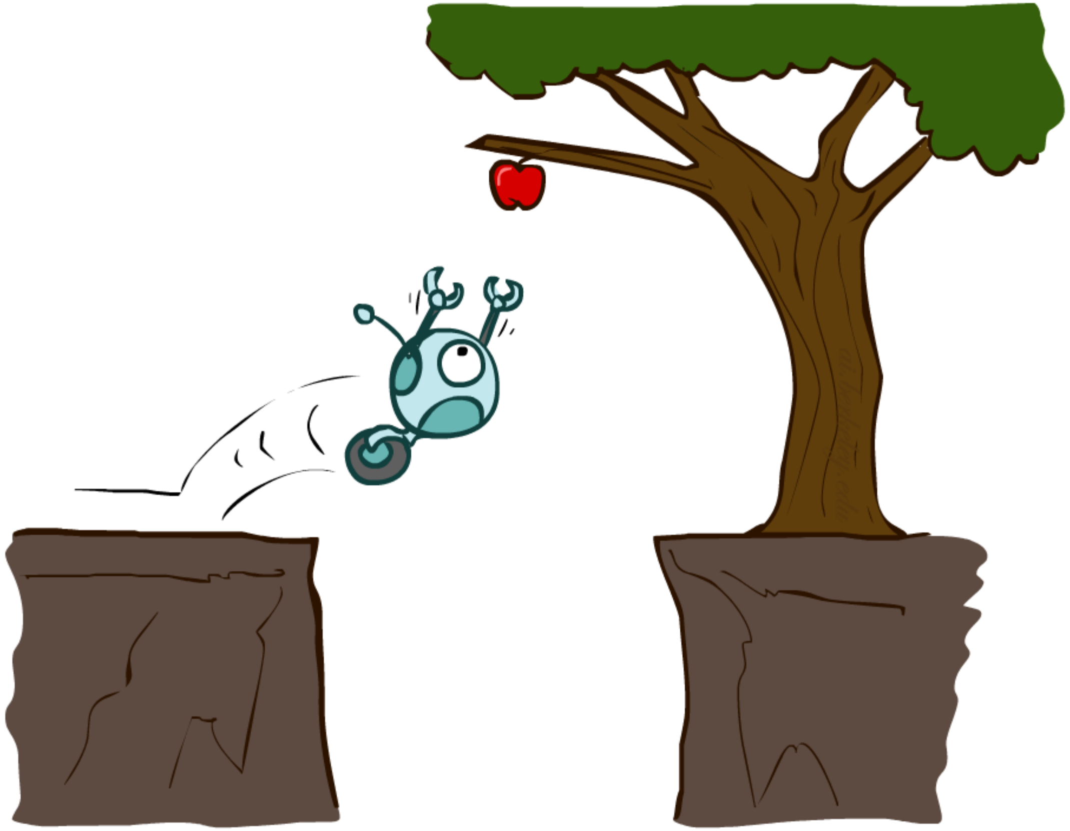


```
function REFLEX-VACUUM-AGENT([location,status]) returns an action

    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

1. Simple reflex agents

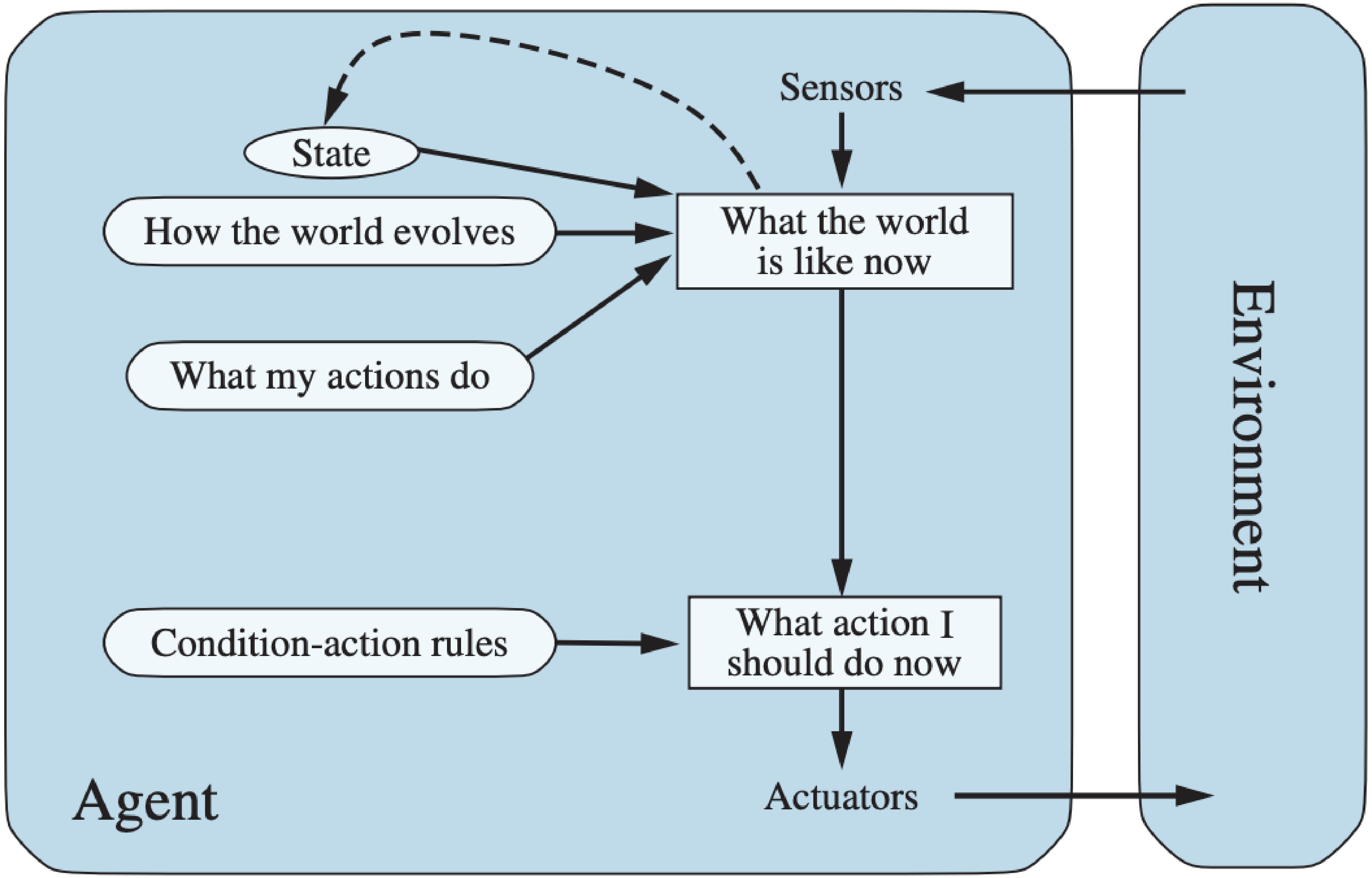
- They are simple to design, easy to implement.
- Their capacities are very limited.
- Work only if the correct decision can be made based on only the current percept: the environment is fully observable



What if the environment is not fully observable?

- In partially unobservable environments, the agent needs to keep track of the currently unobservable part of the world → **model.**
  - Example: the vacuum agent needs to keep track from where it came (left or right).

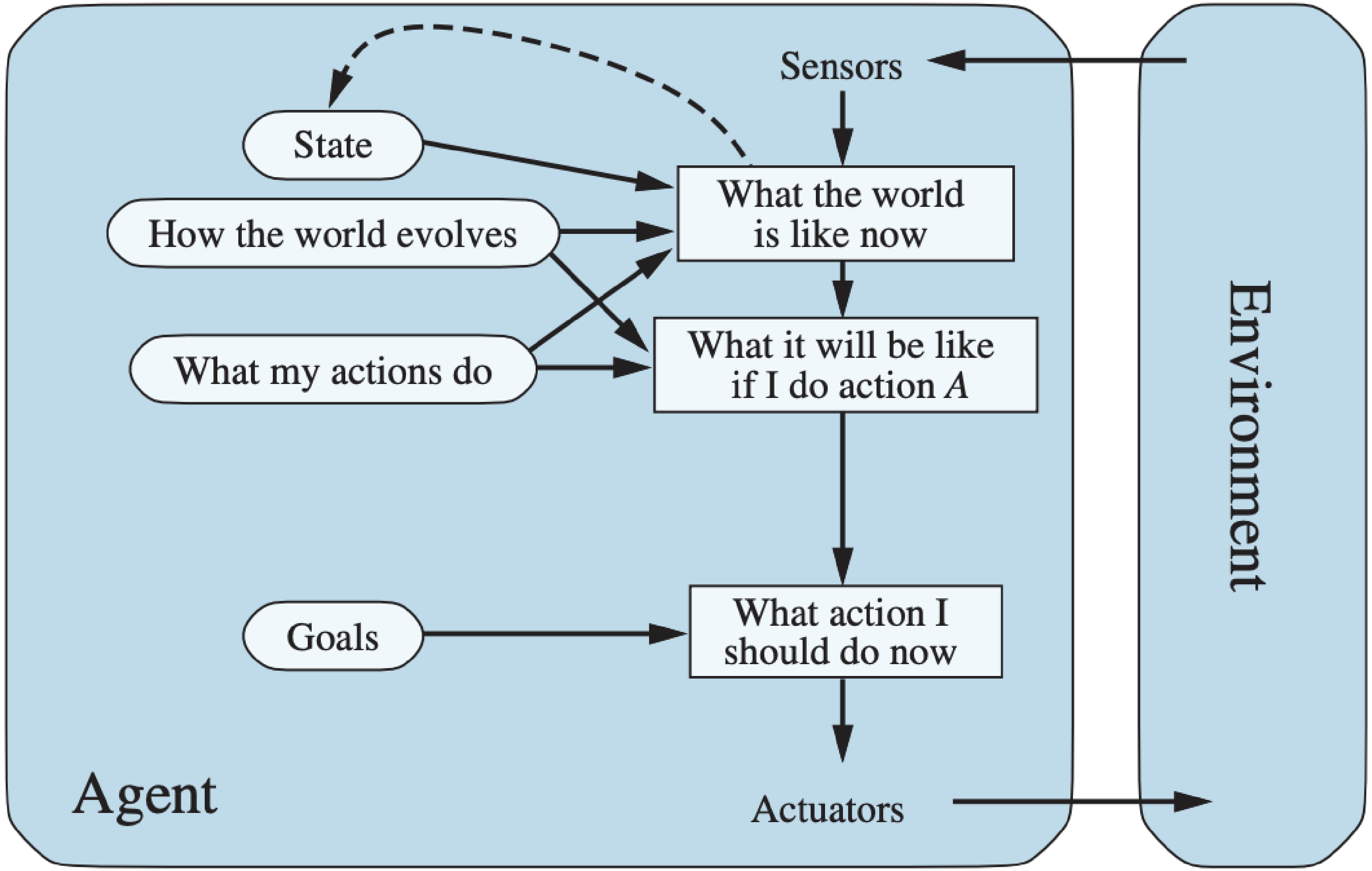
2. Model-based reflex agents



2. Model-based reflex agents

- Internal state: unobserved parts of the world
- To update the internal state, the agent needs:
    - Current percept
    - World model: how the world evolves
    - How the actions of the agent change the world
- Taxi agent, changing lanes:
    - Current view of the road
    - How the unobserved cars would move
    - Turning the steering wheel left causes the vehicle to turn left.
- The unobservable environment is never determined exactly. The agent can only guess.
- Problem: what happened when the taxi agent reaches a junction?

3. Goal-based agents

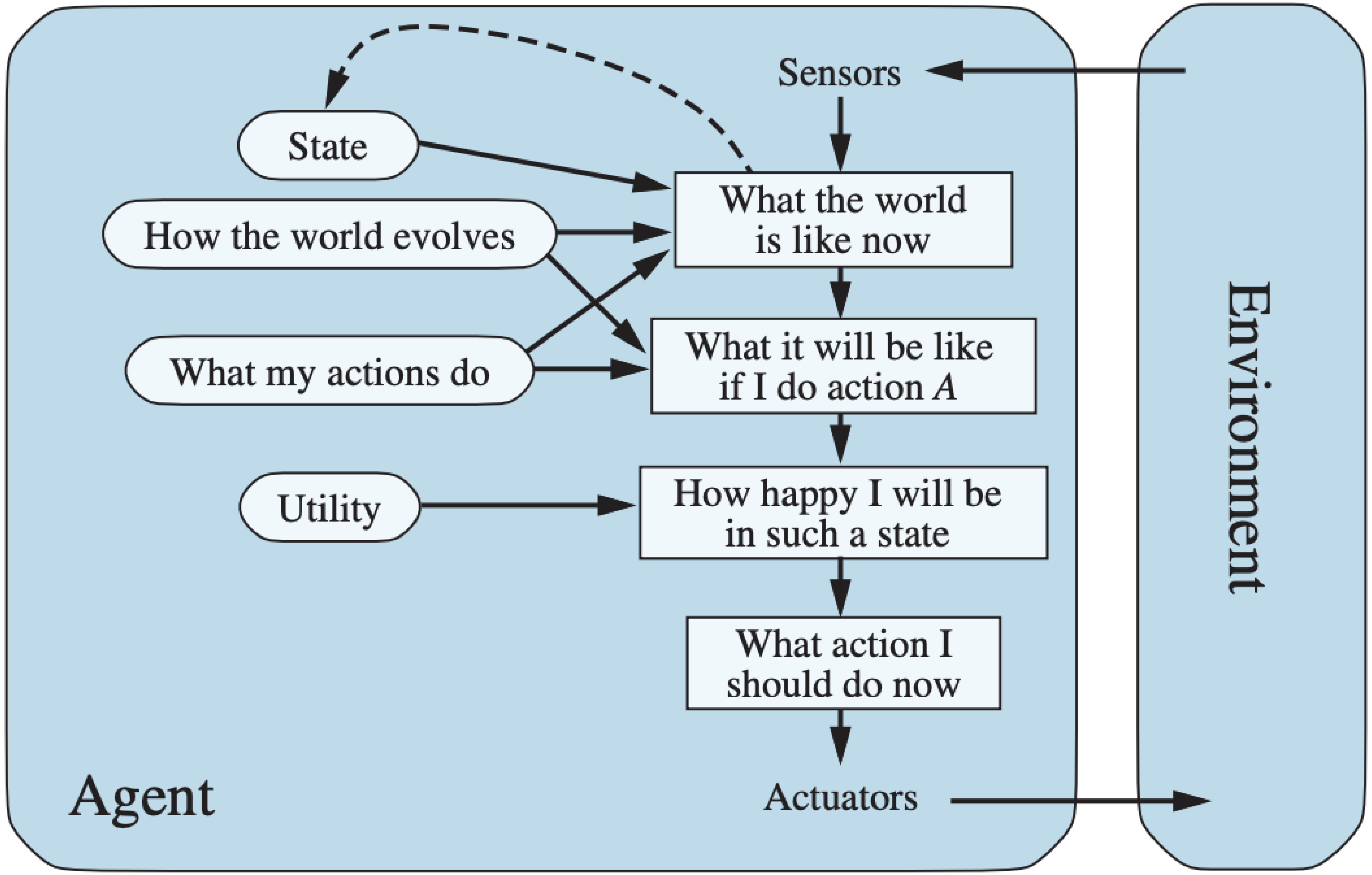

Sensors
State
What the world is like now
How the world evolves
What my actions do
What it will be like if I do action *A*
Goals
What action I should do now
Agent
Actuators
Environment

3. Goal-based agents

- The goal may be reached in one action, but may also need several actions:
    - Consider the future: what will happen if do this action ?
    - Example: when the taxi agent reaches a junction
- Problem: what if the goal can be reached by many ways?
    - Taxi agent:
        - different routes lead to the same place: different solutions.
        - Safety and speed: conflicting goals

    → 4. **utility** (based agents):
        - minimize time and fuel consumption for example.
        - Tradeoff between safety and speed

4. Utility-based agents


Sensors
State
What the world is like now
How the world evolves
What my actions do
What it will be like if I do action *A*
Utility
How happy I will be in such a state
What action I should do now
Actuators
Agent
Environment

4. Utility-based agents

- Goals alone are not enough to generate high-quality behavior in most environments
- Goals just provide a distinction between “happy” and “unhappy” states.
- A more general performance measure allows a comparison of different world states according to exactly how happy they would make the agent
  - Because “happy” does not sound very scientific, economists and computer scientists use the term **utility** instead

- Utility: internal representation of the performance measures.

Goal vs. Utility example: Taxi

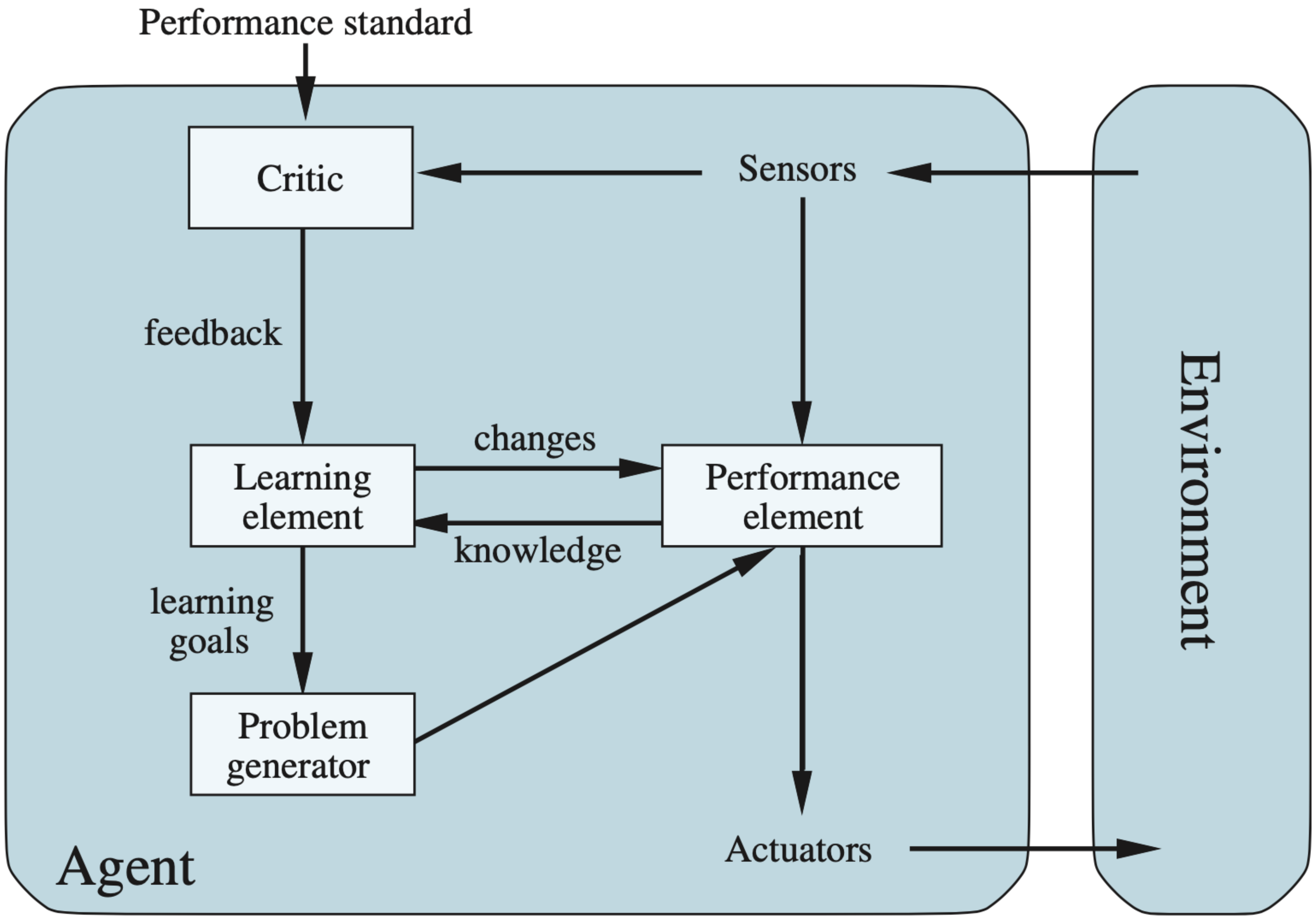
Goal

- agent is tasked with getting from point A to point B. If the agent succeeds, the goal has been satisfied.

Utility

- agent could seek to get from point A to point B in the shortest amount of time, with the minimum expenditure of fuel, or both.

5. Learning Agent



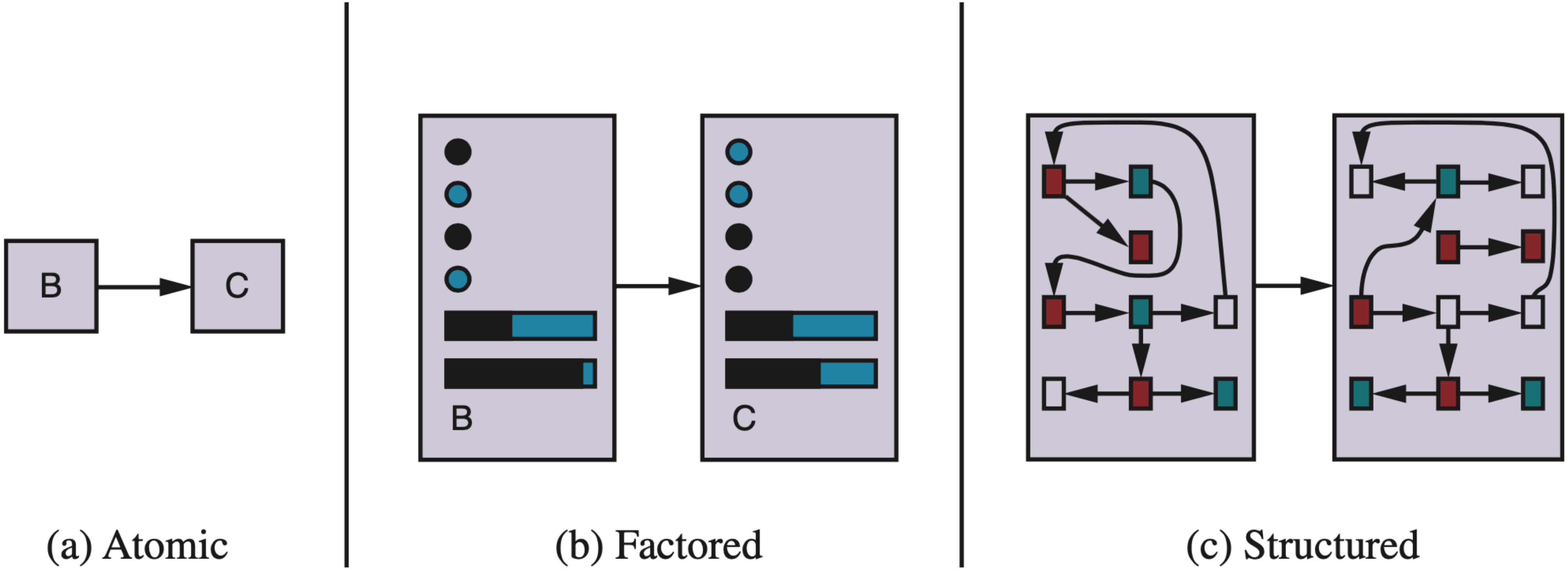
5. Learning Agent

- Learning element: responsible for making improvements
- Performance element: responsible for selecting external actions.
  - The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions
- The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future
- Problem generator: responsible for suggesting actions that will lead to new and informative experiences

5. Learning Agent: Taxi

- Performance element: knowledge and procedures the taxi has for selecting its driving actions
- Critic: observes the world and passes information along to the learning element
  - after the taxi makes a quick left turn across three lanes of traffic, the critic observes the anger of other drivers.
- Learning element formulates a rule saying this was a bad action, and the performance element is modified by installation of the new rule
- Problem generator identifies areas of behavior in need of improvement and suggest experiments
  - trying out the brakes on different road surfaces under different conditions

Environment Representation



(a) Atomic

(b) Factored

(c) Structured

(a) Atomic Representation

- Each state of the world is indivisible—it has no internal structure
- Example: Drive from one city to another. The state is the name of the city
- **Search, game-playing**, **Hidden Markov models**, and **Markov decision processes** all work with atomic representations

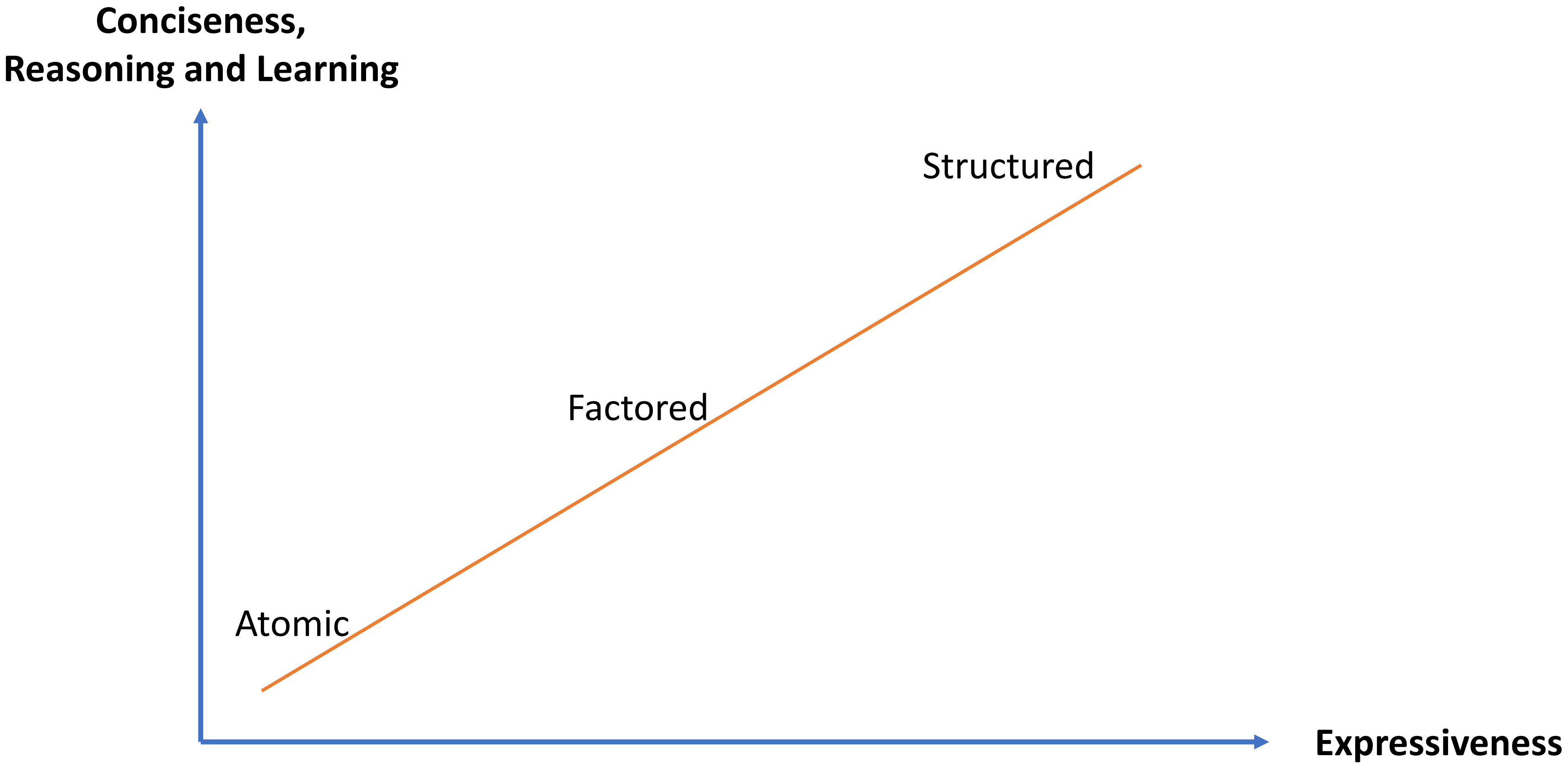
(b) Factored Representation

- Splits up each state into a fixed set of **variables** or **attributes**, each of which can have a **value**.
- Example: Drive from one city to another. Describe the state by current position, how much gas is left, various car indicators (engine, oil, air in wheels …)
- Two different factored states can share some attributes and not others
- **Constraint satisfaction** algorithms, **propositional logic**, **planning**, **Bayesian networks**, and **machine learning**

(c) Structured Representation

- Used when we need to understand the world as having *things* in it that are *related* to each other, not just variables with values
- Example: a large truck is ahead of us and reversing into the driveway of a dairy farm, but a cow has gotten loose and is blocking the truck's path. A factored representation is unlikely to be pre-equipped with the attribute `TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow` with value `true` or `false`.
- **relational databases, first-order logic, first-order probability models, knowledge-based learning, natural language understanding**

Expressiveness vs. Conciseness vs. Reasoning and Learning



How is an Agent different from other software?

- Agents are **autonomous**, that is, they act on behalf of the user

- Agents contain some level of **intelligence**, from fixed rules to learning engines that allow them to adapt to changes in the environment

- Agents don't only act **reactively**, but sometimes also **proactively**

How is an Agent different from other software?

- Agents have **social ability**, that is, they communicate with the user, the system, and other agents as required
- Agents may also **cooperate** with other agents to carry out more complex tasks than they themselves can handle
- Agents may **migrate** from one system to another to access remote resources or even to meet other agents

Summary

- **Agents** interact with **environments** through **actuators** and **sensors**
- The **agent function** describes what the agent does in all circumstances
- The **performance measure** evaluates the environment sequence
- A **perfectly rational** agent maximizes **expected** performance
- Agent programs implement agent functions
- **PEAS** descriptions define **task environments**
- Environments are categorized along several dimensions:
  - **observable? deterministic? episodic? static? discrete? single-agent?**
- Several basic agent architectures exist:
  - **reflex, model-based reflex, goal-based, utility-based**