

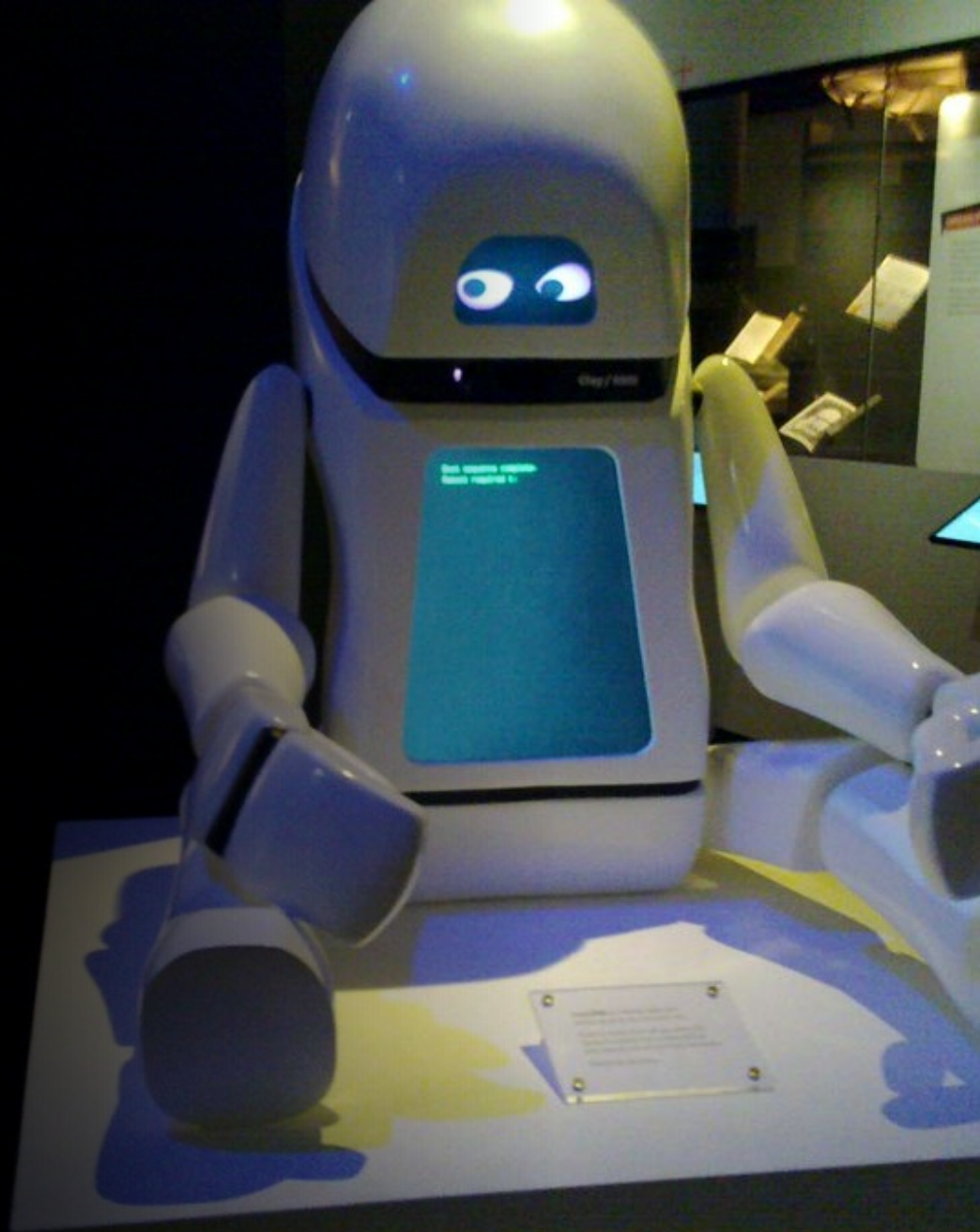


# CS 5/7320 Artificial Intelligence

## Intelligent Agents AIMA Chapter 2

---

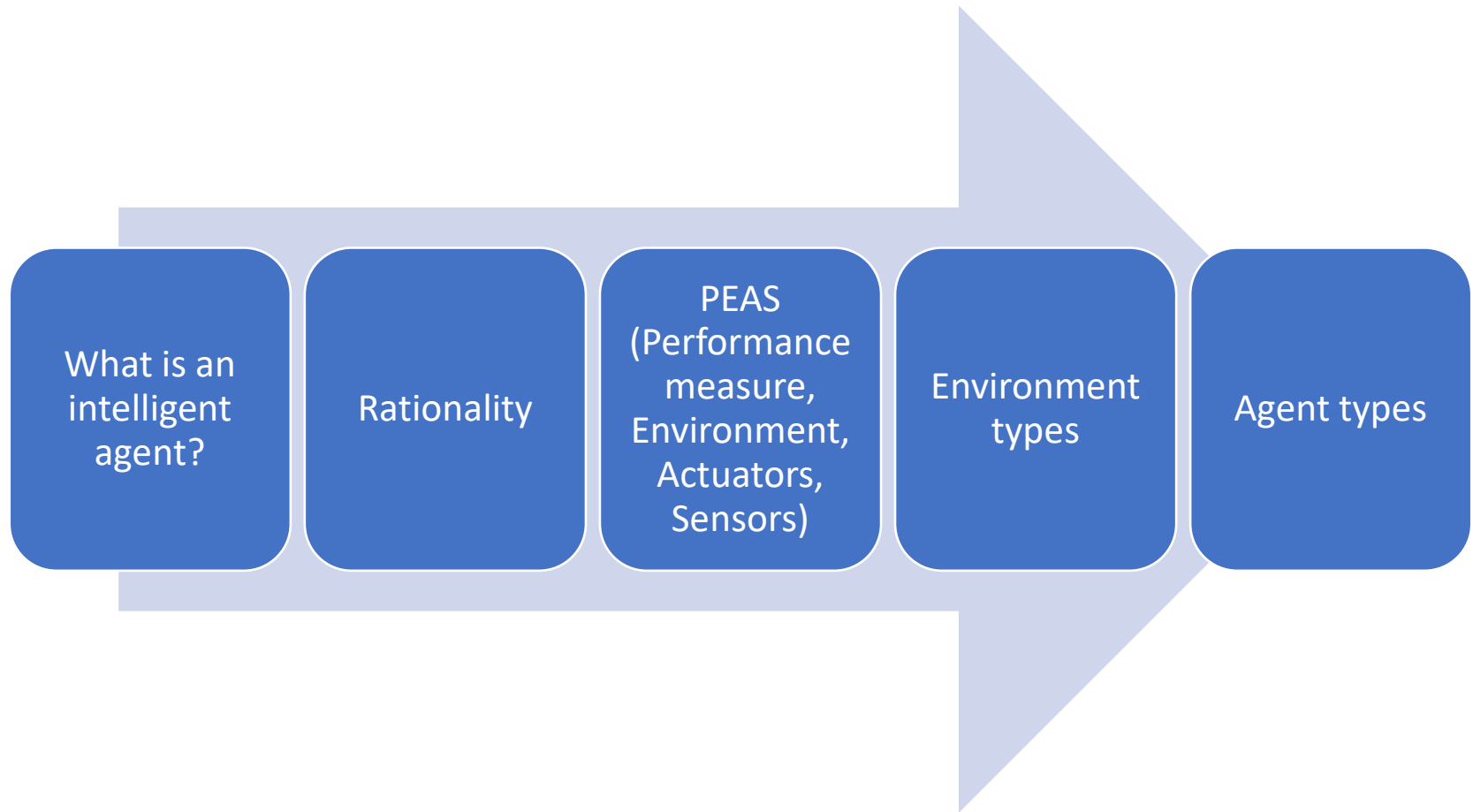
Slides by Michael Hahsler  
based on slides by Svetlana Lazepnik  
with figures from the AIMA textbook.



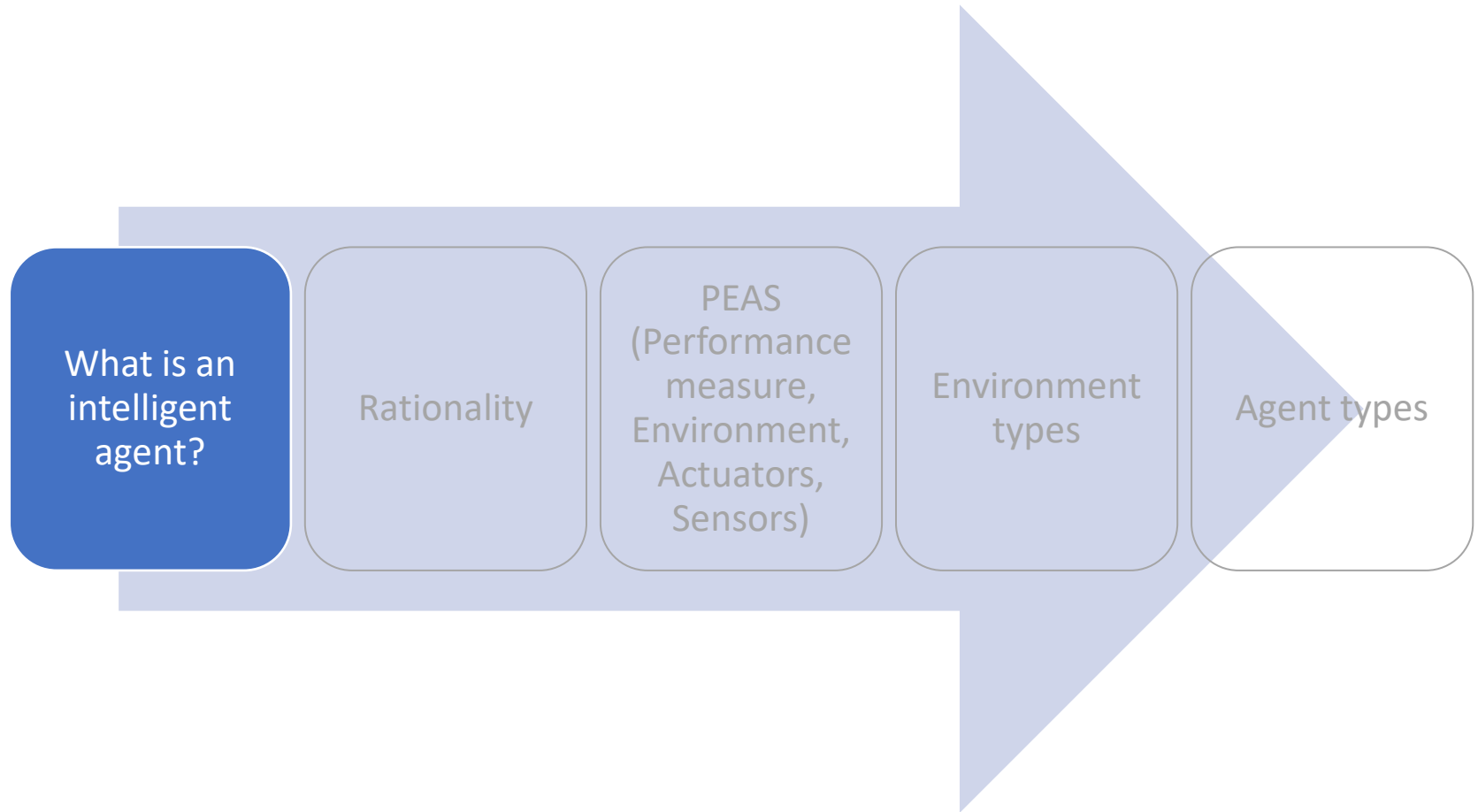
This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Image: "Robot at the British Library Science Fiction Exhibition"  
by BadgerGravling

# Outline

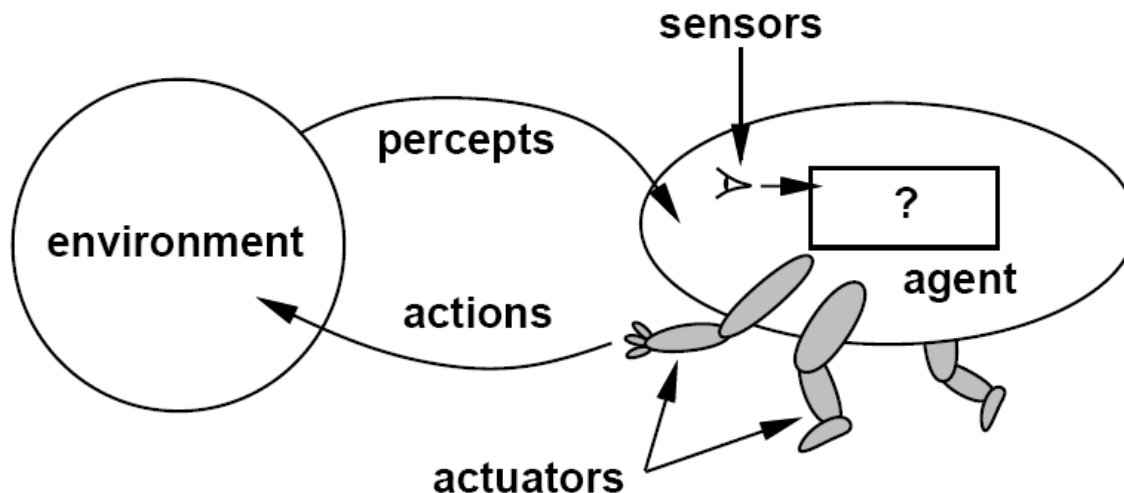


# Outline



# What is an Agents?

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

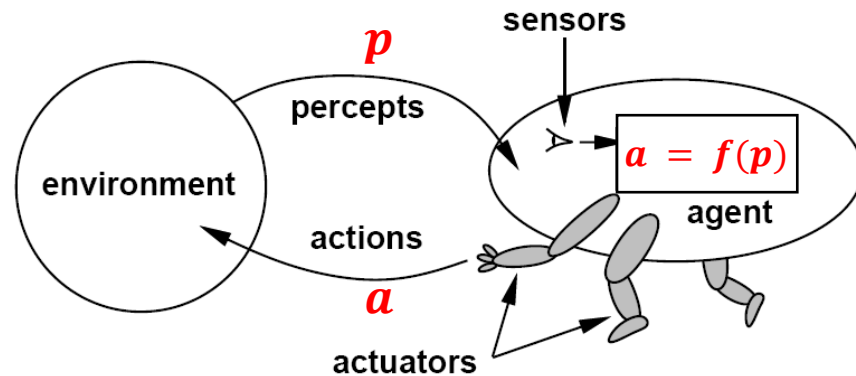


- **Control theory:** A **closed-loop control system** (= feedback control system) is a set of mechanical or electronic devices that automatically regulate a process variable to a desired state or set point without human interaction. The agent is called a controller.
- **Softbot:** Agent is a software program that runs on a host device.

# Agent Function and Agent Program

The **agent function** maps from the set of all possible *percept sequences*  $P^*$  to the *set of actions*  $A$  formulated as an abstract mathematical function.

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



The **agent program** is a concrete implementation of this function for a given physical system.

Agent = architecture (hardware) + agent program (implementation of  $f$ )



- Sensors
- Memory
- Computational power

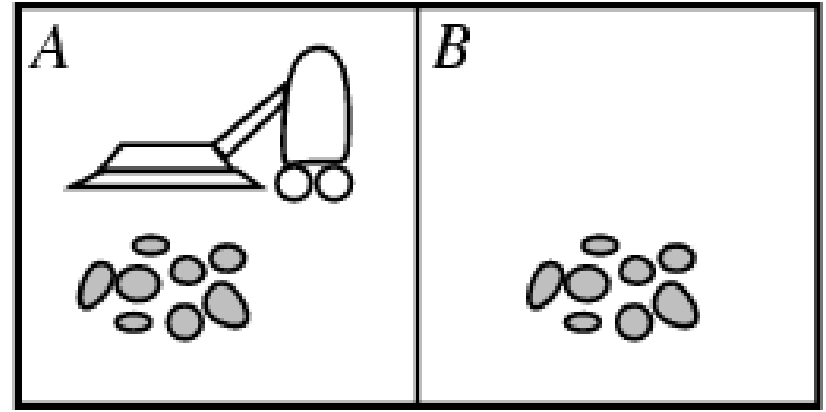
# Example: Vacuum-cleaner World

- **Percepts:**

Location and status,  
e.g., [A, Dirty]

- **Actions:**

Left, Right, Suck, NoOp



Most recent  
Percept  $p$

Agent function:  $f : P^* \rightarrow A$

Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
...	
[A, Clean], [B, Clean]	Left
...	
[A, Clean], [B, Clean], [A, Dirty]	Suck
...	

Implemented agent program:

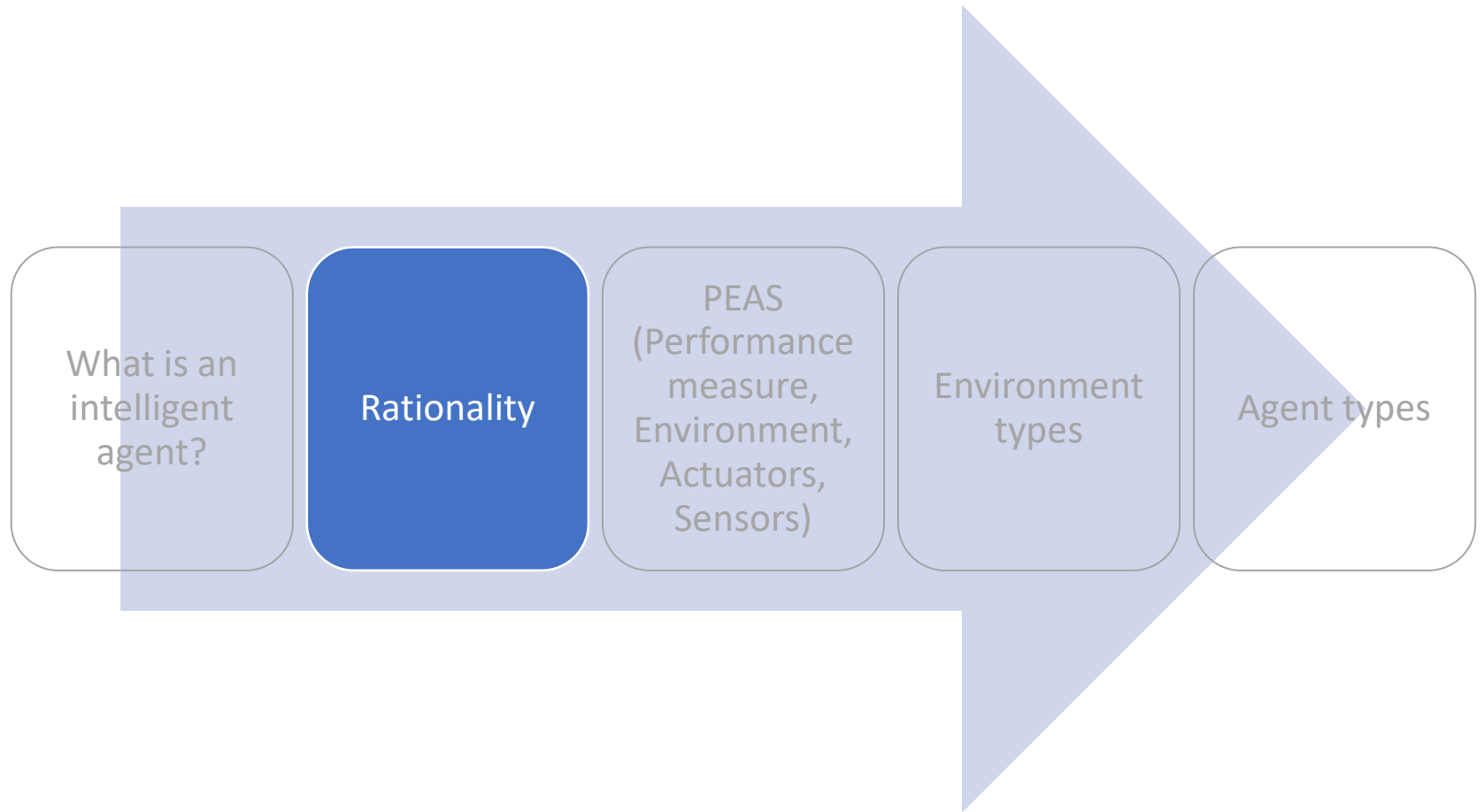
**function** Vacuum-Agent([location, status])

returns an **action**  $a$

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

**Problem:** This table can become infinitively large!

# Outline



# Rational Agents: What is Good Behavior?

## Foundation

- **Consequentialism:** Evaluate behavior by its consequences.
- **Utilitarianism:** Maximize happiness and well-being.

## Definition of a rational agent:

*“For each possible percept sequence, a rational agent should select an **action** that **maximizes its expected performance measure**, given the evidence provided by the **percept sequence** and the **agent’s built-in knowledge**.”*

- **Performance measure:** An *objective* criterion for success of an agent's behavior (often called utility function or reward function).
- **Expectation:** Outcome averaged over all possible situations that may arise.

**Rule:** Pick the action that maximize the expected utility

$$a = \operatorname{argmax}_{a \in A} E(U \mid a)$$



# Rational Agents

**Rule:** Pick the action that maximize the expected utility

$$a = \operatorname{argmax}_{a \in A} E(U \mid a)$$

This means:

- **Rationality is an ideal** – it implies that no one can build a better agent
- **Rationality  $\neq$  Omniscience** – rational agents can make mistakes if percepts and knowledge do not suffice to make a good decision
- **Rationality  $\neq$  Perfection** – rational agents maximize **expected** outcomes not actual outcomes
- **It is rational to explore and learn** – i.e., use **percepts** to supplement prior knowledge and become autonomous
- **Rationality is often bounded** by available memory, computational power, available sensors, etc.

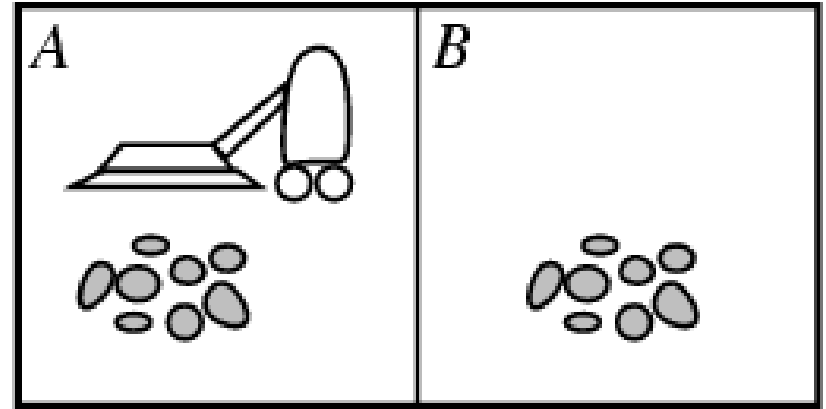
# Example: Vacuum-cleaner World

- **Percepts:**

Location and status,  
e.g., [A, Dirty]

- **Actions:**

Left, Right, Suck, NoOp



Agent function:

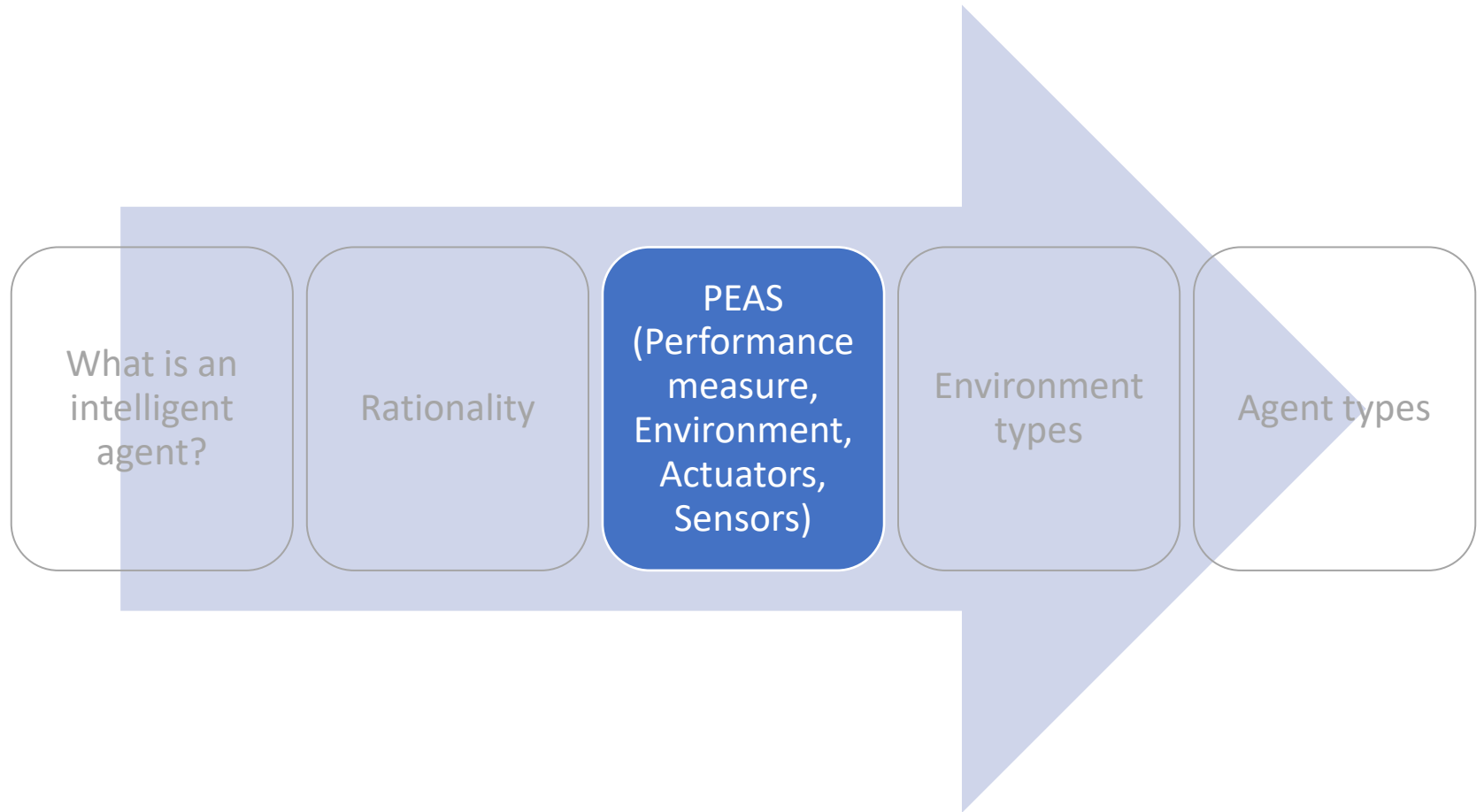
Percept Sequence	Action
[A, Clean]	Right
[A, Dirty]	Suck
...	
[A, Clean], [B, Clean]	Left
...	

Implemented agent program:

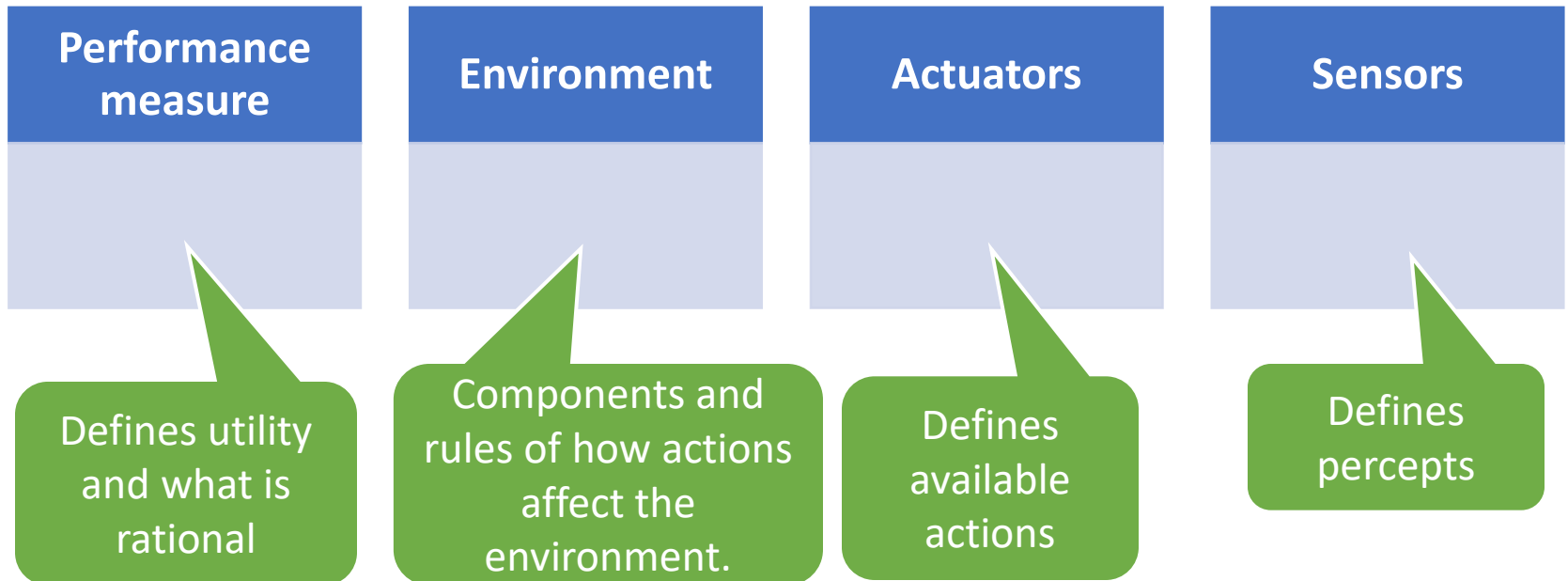
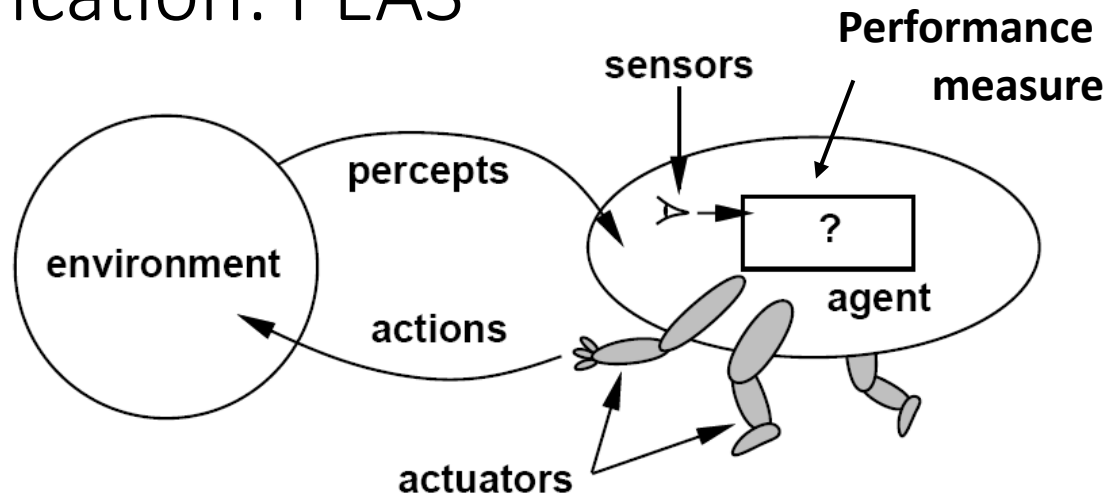
```
function Vacuum-Agent([location, status])  
    returns an action  
  
    if status = Dirty then return Suck  
    else if location = A then return Right  
    else if location = B then return Left
```

**What could be a performance measure?**  
**Is this agent program rational?**

# Outline



# Problem Specification: PEAS



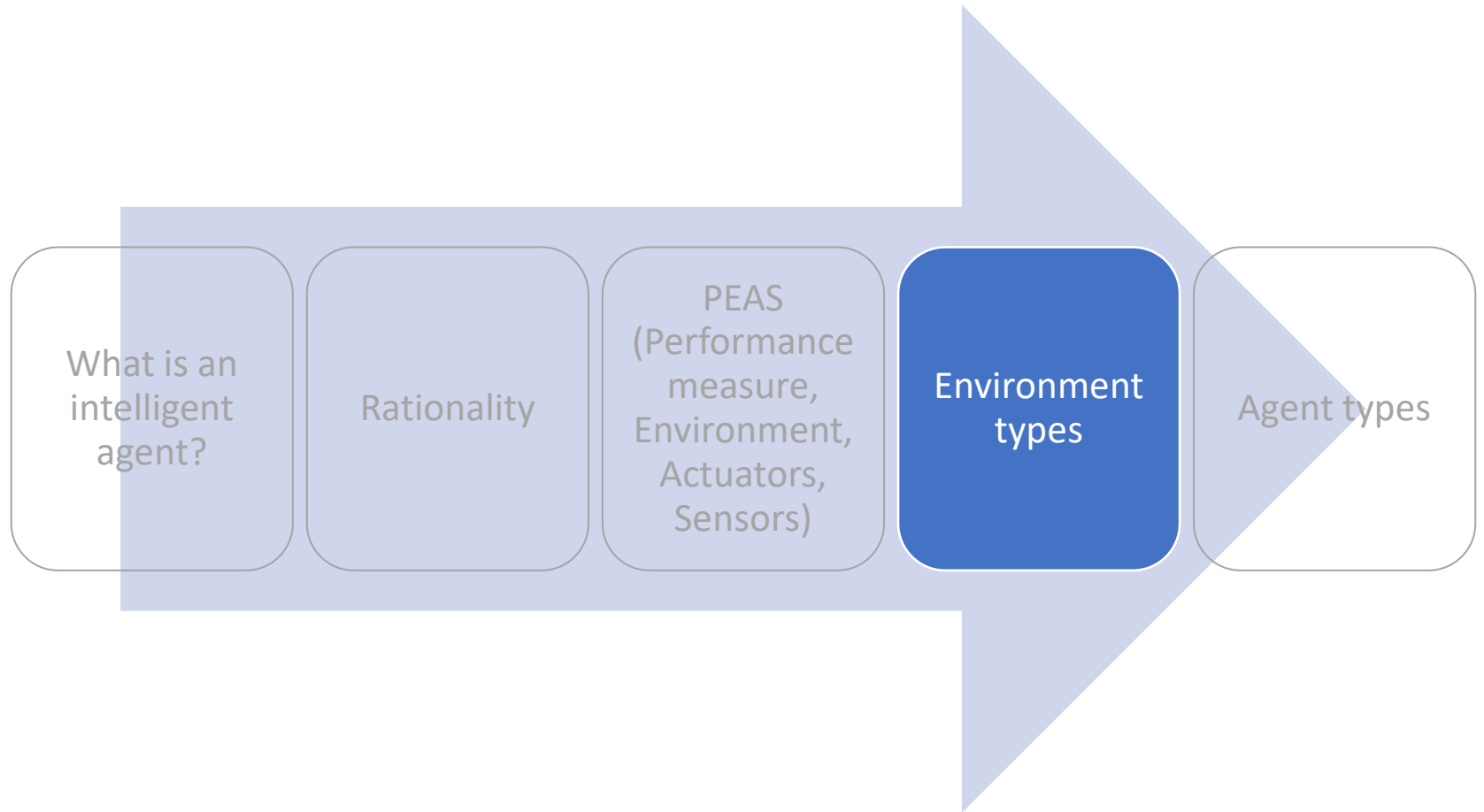
# Example: Automated Taxi Driver

Performance measure	Environment	Actuators	Sensors
<ul style="list-style-type: none"><li>• Safe</li><li>• fast</li><li>• legal</li><li>• comfortable trip</li><li>• maximize profits</li></ul>	<ul style="list-style-type: none"><li>• Roads</li><li>• other traffic</li><li>• pedestrians</li><li>• customers</li></ul>	<ul style="list-style-type: none"><li>• Steering wheel</li><li>• accelerator</li><li>• brake</li><li>• signal</li><li>• horn</li></ul>	<ul style="list-style-type: none"><li>• Cameras</li><li>• sonar</li><li>• speedometer</li><li>• GPS</li><li>• Odometer</li><li>• engine sensors</li><li>• keyboard</li></ul>

# Example: Spam Filter

Performance measure	Environment	Actuators	Sensors
<ul style="list-style-type: none"><li>• Accuracy: Minimizing false positives, false negatives</li></ul>	<ul style="list-style-type: none"><li>• A user's email account</li><li>• email server</li></ul>	<ul style="list-style-type: none"><li>• Mark as spam</li><li>• delete</li><li>• etc.</li></ul>	<ul style="list-style-type: none"><li>• Incoming messages</li><li>• other information about user's account</li></ul>

# Outline



# Environment Types

**Fully observable:** The agent's sensors give it access to the complete state of the environment. The agent can “see” the whole environment.

**VS.**

**Partially observable:** The agent cannot see all aspects of the environment. E.g., it can't see through walls

**Deterministic:** Changes in the environment is completely determined by the current state of the environment and the agent's action.

**VS.**

**Stochastic:** Changes cannot be determined from the current state and the action (there is some randomness).

**Strategic:** The environment is stochastic and adversarial. It chooses actions strategically to harm the agent. E.g., a game where the other player is modeled as part of the environment.

**Known:** The agent knows the rules of the environment and can predict the outcome of actions.

**VS.**

**Unknown:** The agent cannot predict the outcome of actions.



# Environment Types

**Static:** The environment is **not** changing while agent is deliberating.

**Semidynamic:** the environment is static, but the agent's performance score depends on how fast it acts.

**Discrete:** The environment provides a fixed number of distinct percepts, actions, and environment states. Time can also evolve in a discrete or continuous fashion.

**Episodic:** Episode = a self-contained sequence of actions. The agent's choice of action in one episode does not affect the next episodes. The agent does the same task repeatedly.

**Single agent:** An agent operating by itself in an environment.

**vs.**

**Dynamic:** The environment is changing while the agent is deliberating.

**vs.**

**Continuous:** Percepts, actions, state variables or time are continuous leading to an infinite state, percept or action space.

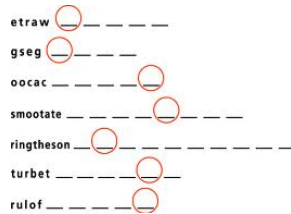
**vs.**

**Sequential:** Actions now affect the outcomes later. E.g., learning makes problems sequential.

**vs.**

**Multi-agent:** Agent cooperate or compete in the same environment.

# Examples of Different Environments



Word jumble  
solver



Chess with  
a clock



Scrabble

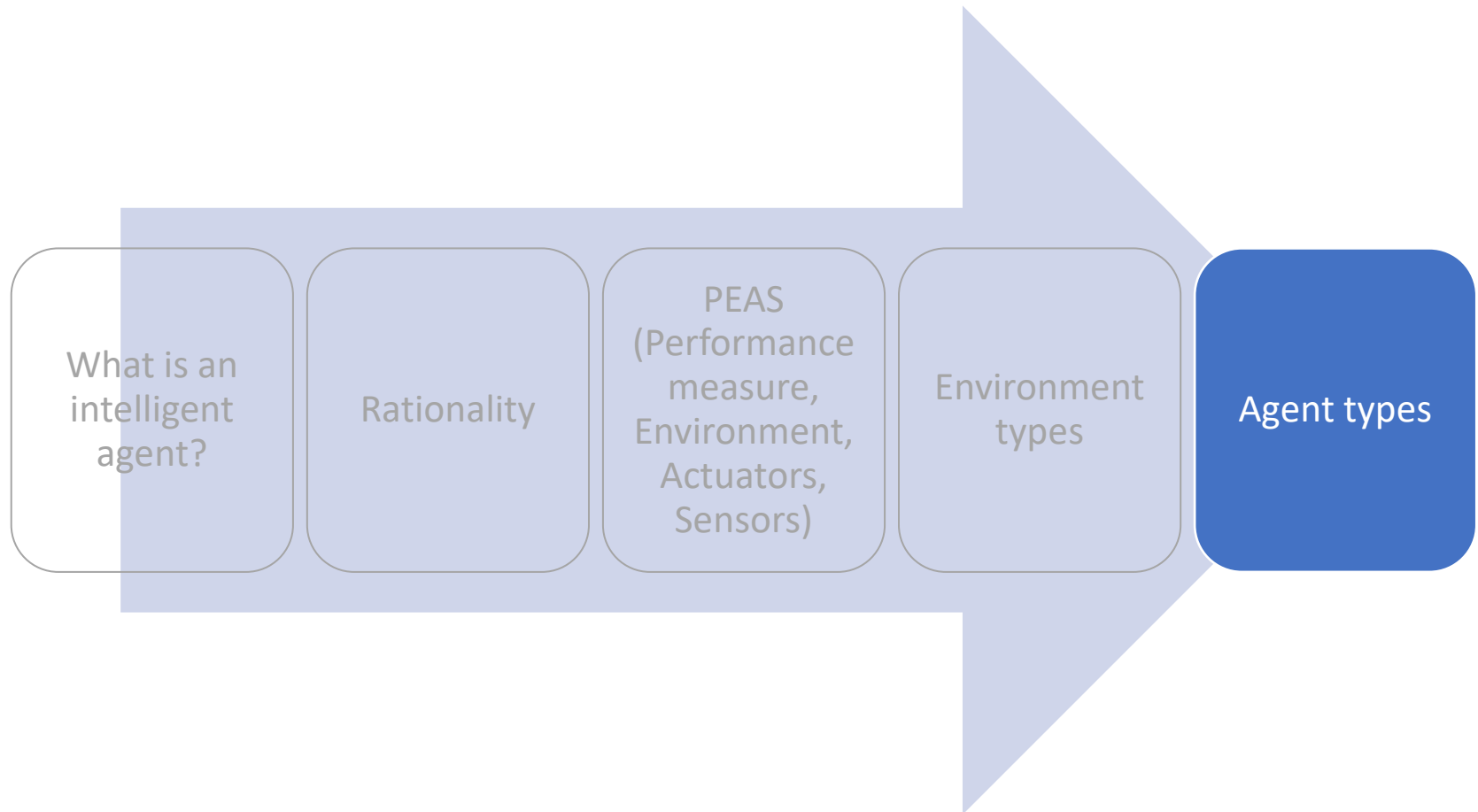


Taxi driving

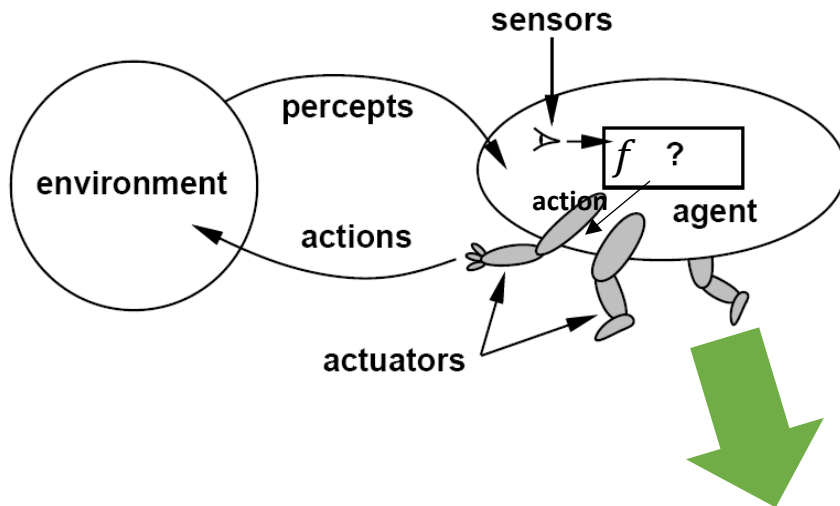
Observable	Fully	Fully	Partially	Partially
Deterministic	Deterministic	Determ. game Mechanics + Strategic*	Stochastic +Strategic	Stochastic
Episodic?	Episodic	Episodic	Episodic	Sequential
Static	Static	Semidynamic	Static	Dynamic
Discrete	Discrete	Discrete	Discrete	Continuous
Single agent	Single	Multi*	Multi*	Multi*

\* Can be models as a single agent problem with the other agent(s) in the environment.

# Outline



# Designing a Rational Agent

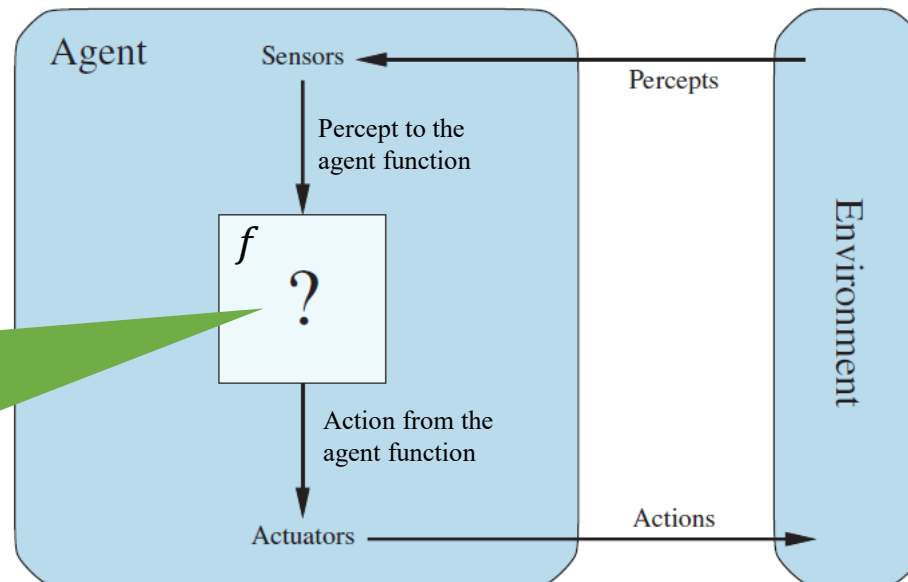


Remember the definition of a rational agent:

*“For each possible percept sequence, a rational agent should select an **action** that **maximizes its expected performance measure**, given the evidence provided by the **percept sequence** and the **agent’s built-in knowledge**.”*

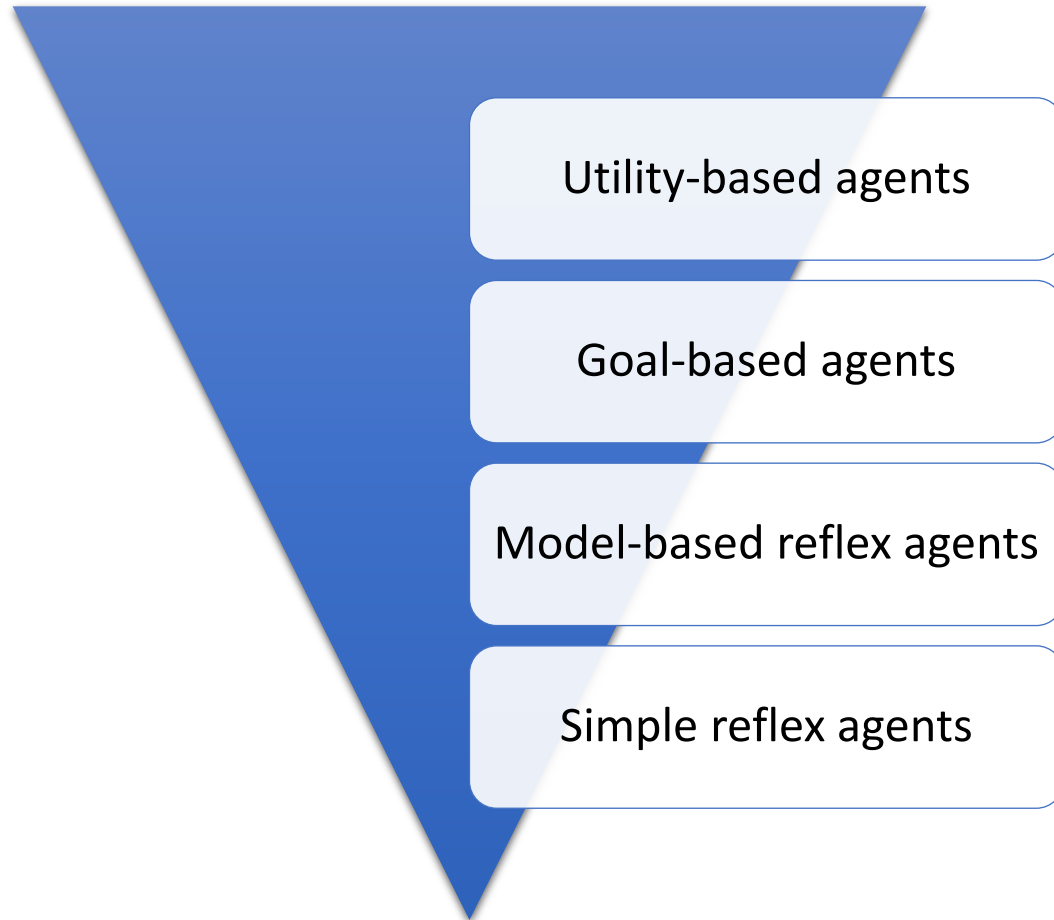
## Agent Function

- Assess performance measure
- Remember percept sequence
- Built-in knowledge



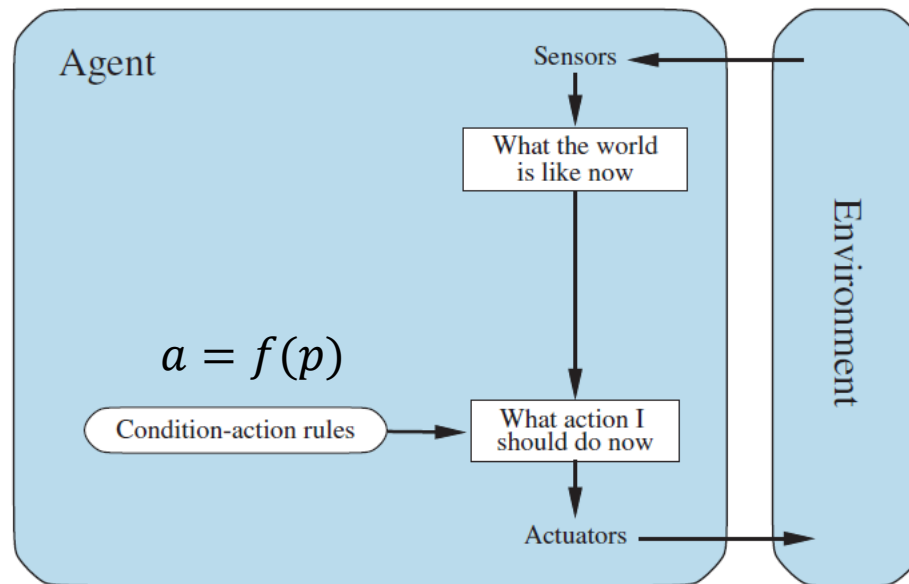
**Note:** Everything outside the agent function can be seen as the environment.

# Hierarchy of Agent Types



# Simple Reflex Agent

- Uses only built-in knowledge in the form of **rules** that select action only **based on the current percept**. This is typically very fast!
- The **agent does not know about the performance measure**! But well-designed rules can lead to good performance.
- The agent needs no memory and ignores all past percepts.

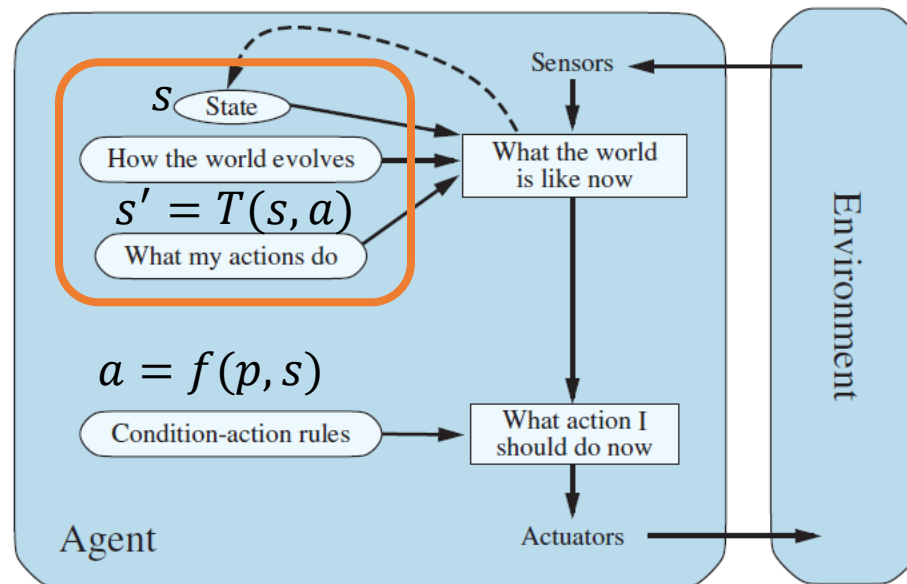


The interaction is a sequence:  $p_0, a_0, p_1, a_1, p_2, a_2, \dots, p_t, a_t, \dots$

**Example:** A simple vacuum cleaner that uses rules based on its current sensor input.

# Model-based Reflex Agent

- Maintains a **state variable** to keep track of aspects of the environment that cannot be currently observed. I.e., it has memory and knows how the environment reacts to actions (called **transition function**).
- The state is updated using the percept.
- There is now more information for the **rules** to make better decisions.

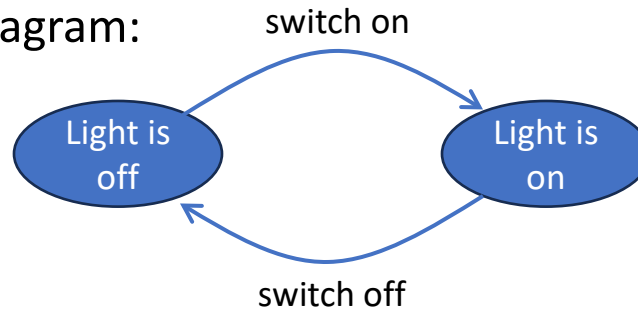


The interaction is a sequence:  $s_0, a_0, p_1, s_1, a_1, p_2, s_2, a_2, p_3, \dots, p_t, s_t, a_t, \dots$

**Example:** A vacuum cleaner that remembers where it has already cleaned.

# Transition Function

- The environment is modeled as a discrete **dynamical system**.
- Changed in the environment are a sequence of states  $s_0, s_1, \dots, s_T$ , where the index is the time step.
- Example of a state diagram:



- States change because of:
  - a. System dynamics of the environment.
  - b. The actions of the agent.
- Both types of changes are represented by the transition function written as

$$T: S \times A \rightarrow S$$

or

$$s' = T(s, a)$$

$S$  ... set of states

$A$  ... set of available actions

$a \in A$  ... an action

$s \in S$  ... current state

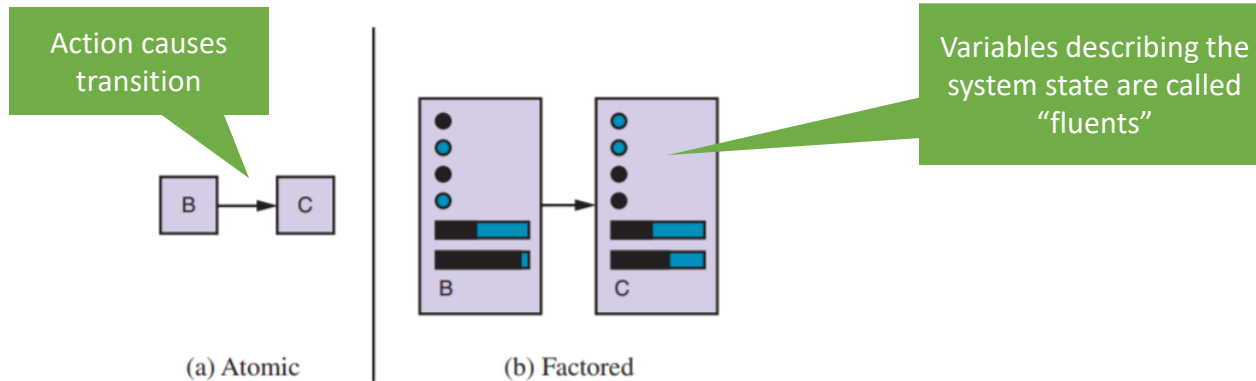
$s' \in S$  ... next state



# State Representation

States help to keep track of the environment and the agent in the environment. This is often also called the **system state**. The representation can be

- **Atomic**: Just a label for a black box. E.g., A, B
- **Factored**: A set of attribute values called fluents.  
E.g., [location = left, status = clean, temperature = 75 deg. F]



We often construct atomic labels from factored information. E.g.: If the agent's state is the coordinate  $x = 7$  and  $y = 3$ , then the atomic state label could be the string  $"(7, 3)"$ . With the atomic representation, we can only compare if two labels are the same. With the factored state representation, we can reason more and calculate the distance between states!

The set of all possible states is called the **state space  $S$** . This set is typically very large!

# Old-school vs. Smart Thermostat



Old-school thermostat

**Percepts**

**States**



Smart thermostat

**Percepts**

**States**

# Old-school vs. Smart Thermostat



Set  
temperature  
range



Change  
temperature  
when you  
are too  
cold/warm.

## Old-school thermostat

### Percepts

temperature:  
Low, ok, high

### States

No states need

## Smart thermostat

### Percepts

- Temp: deg. F
- Outside temp.
- Weather report
- Energy curtailment
- Someone walking by
- Someone changes temp.
- Day & time
- ...

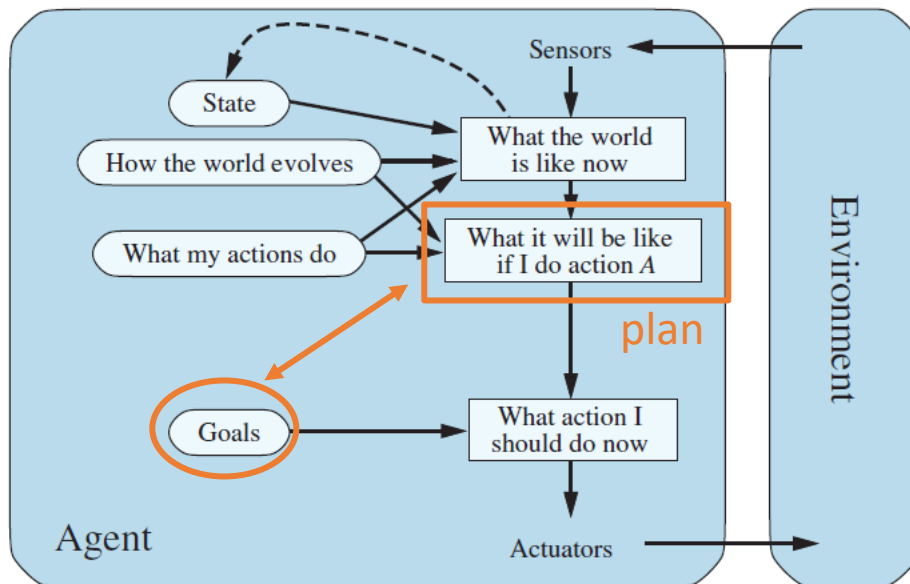
### States

Factored states

- Estimated time to cool the house
- Someone home?
- How long till someone is coming home?
- A/C: on, off

# Goal-based Agent

- The agent has the task to reach a defined **goal state** and is then finished.
- The agent needs to move towards the goal. It can use **search algorithms** to plan actions that lead to the goal.
- Performance measure: the **cost to reach the goal**.



$$a = \operatorname{argmin}_{a_0 \in A} \left[ \underbrace{\sum_{t=0}^T c_t}_{\text{Sum of the cost of a planned sequence of actions that leads to a goal state}} \mid s_T \in S^{\text{goal}} \right]$$

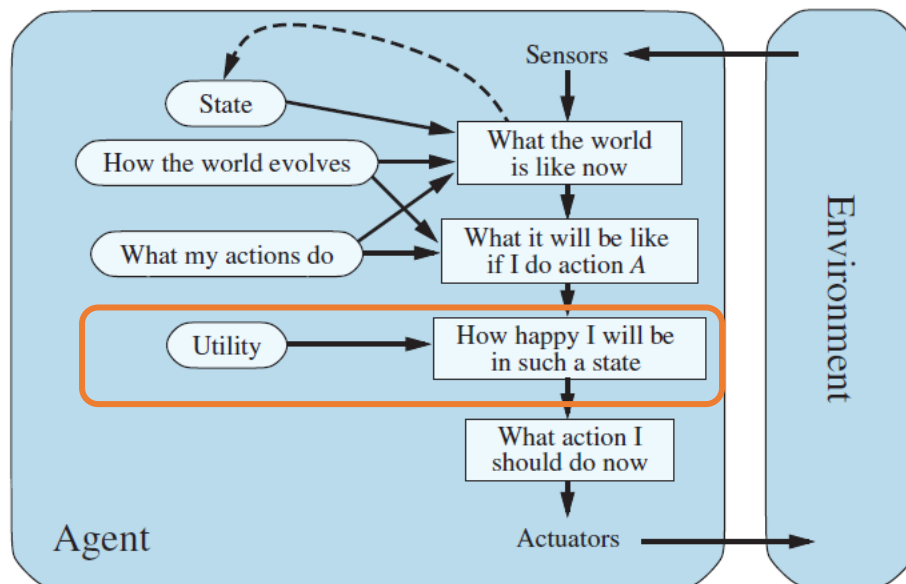
The interaction is a sequence:  $s_0, a_0, p_1, s_1, a_1, p_2, s_2, a_2, \dots, s^{\text{goal}}$

cost

**Example:** Solving a puzzle. What action gets me closer to the solution?

# Utility-based Agent

- The agent uses a utility function to evaluate the **desirability of each possible states**. This is typically expressed as the reward of being in a state  $R(s)$ .
- Choose actions to stay in desirable states.
- Performance measure: The discounted sum of **expected utility over time**.



$$a = \operatorname{argmax}_{a_0 \in A} \mathbb{E} \left[ \underbrace{\sum_{t=0}^{\infty} \gamma^t r_t}_{\text{Utility is the expected future discounted reward}} \right]$$

Utility is the  
expected future  
discounted reward

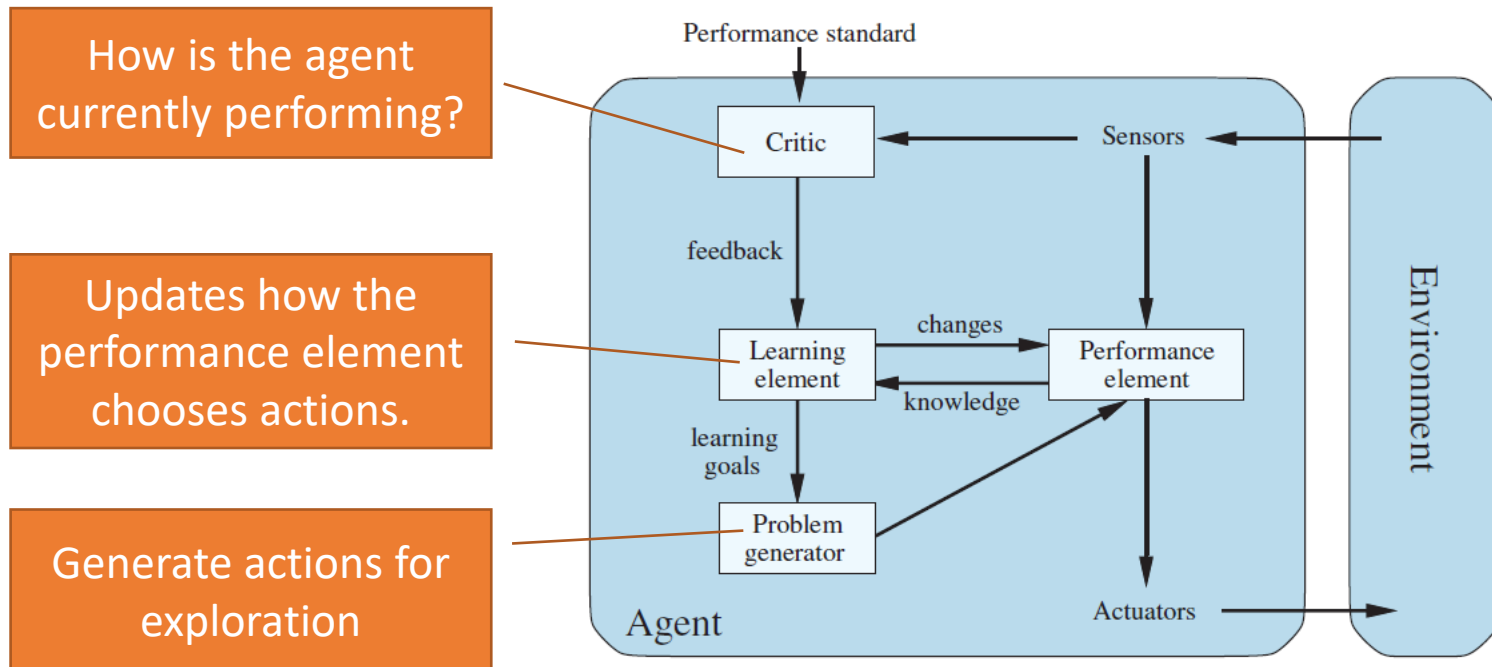
**Techniques:** Markov decision  
processes, reinforcement learning

The interaction is a sequence:  $s_0, a_0, \underbrace{p_1, s_1, a_1, p_2, s_2, a_2, \dots}_{\text{reward}}$

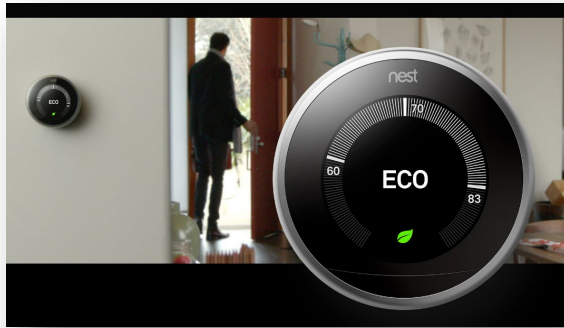
**Example:** An autonomous Mars rover prefers states where its battery is not critically low.

# Agents that Learn

The **learning element** modifies the agent program (reflex-based, goal-based, or utility-based) to improve its performance.



# Example: Smart Thermostat



Change temperature when you are too cold/warm.

Reflex Agent?

Goal-based?

Utility-based?

## Smart thermostat

### Percepts

- Temp: deg. F
- Outside temp.
- Weather report
- Energy curtailment
- Someone walking by
- Someone changes temp.
- Day & time
- ...

### States

#### Factored states

- Estimated time to cool the house
- Someone home?
- How long till someone is coming home?
- A/C: on, off

# Example: Modern Vacuum Robot

Features are:

- Control via App
- Cleaning Modes
- Navigation
- Mapping
- Boundary blockers



iRobot's Roomba brand has become as synonymous with robot vacuum as Q-tips is with cotton swabs. The Wi-Fi-enabled Roomba 960 is ample evidence why. It turns a tiresome chore into something you can almost look forward to. With three cleaning modes and dirt-detecting sensors, it kept all the floor surfaces in our testing immaculate, and its camera-driven navigation and mapping were superb. Its easy-to-use app provides alerts and detailed cleaning reports. The ability to control it with Amazon Alexa and Google Home voice commands are just the cherry on top.

Source: <https://www.techhive.com/article/3269782/best-robot-vacuum-cleaners.html>



# PEAS Description of a Modern Robot Vacuum



**Performance  
measure**

**Environment**

**Actuators**

**Sensors**

# What Type of Intelligent Agent is a Modern Robot Vacuum?



Is it learning?

Utility-based agents

Goal-based agents

Model-based reflex agents

Simple reflex agents

Does it collect utility over time? How would the utility for each state be defined?

Does it have a goal state?

Does it store state information. How would they be defined (atomic/factored)?

Does it use simple rules based on the current percepts?



Check what applies

# What Type of Intelligent Agent is this?



Default (GPT-3.5)



the sun is shining. It is



a beautiful day. The warm rays of sunlight illuminate the surroundings, creating a cheerful and inviting atmosphere. People are out and about, enjoying outdoor activities, and the clear blue sky adds to the overall sense of positivity. Whether you're taking a leisurely stroll, having a picnic, or simply basking in the sun's glow, it's a perfect time to appreciate the natural beauty and soak in the uplifting energy that the sunshine brings.

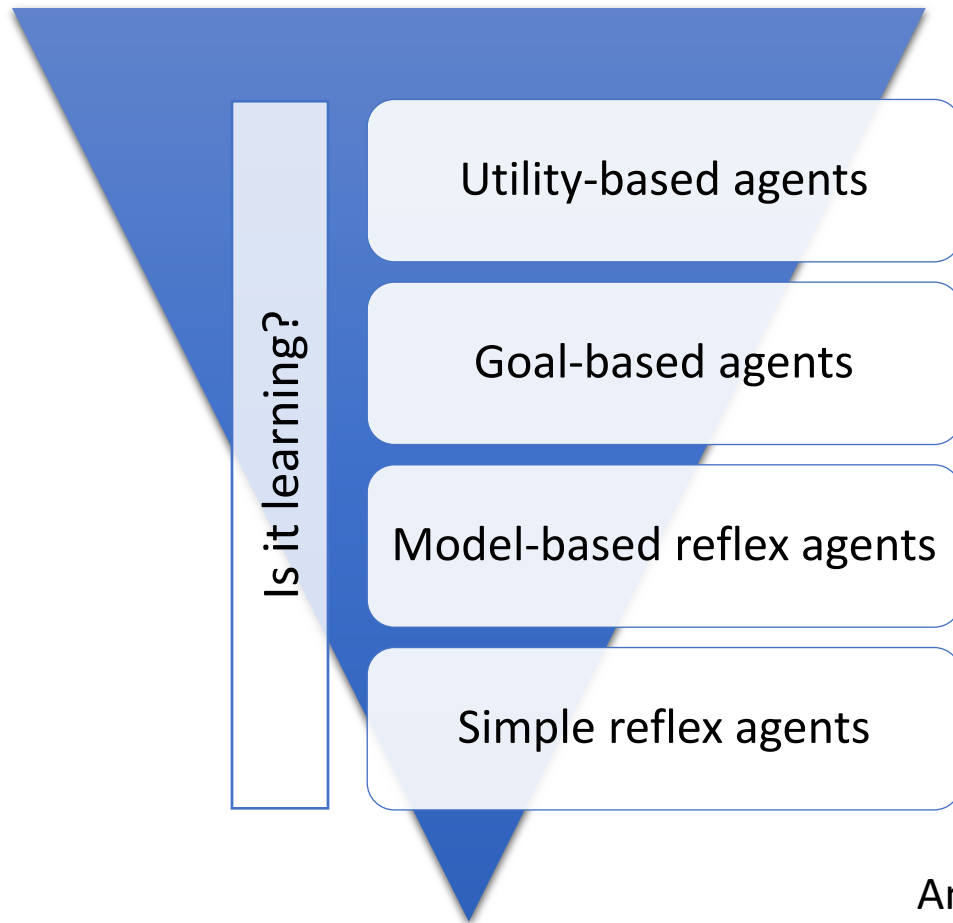


# PEAS Description of ChatGPT



Performance measure	Environment	Actuators	Sensors

# What Type of Intelligent Agent is ChatGPT?



Does it collect utility over time? How would the utility for each state be defined?

Does it have a goal state?

Does it store state information. How would they be defined (atomic/factored)?

Does it use simple rules based on the current percepts?



Check what applies

Answer the following questions:

- Does ChatGPT pass the Turing test?
- Is ChatGPT a rational agent? Why?

We will talk about knowledge-based agents later.

# Intelligent Systems a Sets of Agents: Self-driving Car



It should learn!

Utility-based agents

Goal-based agents

Model-based reflex agents

Simple reflex agents

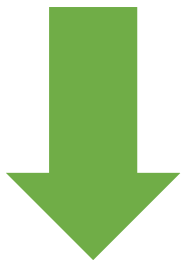
Make sure the passenger has a pleasant drive  
(not too much sudden breaking = utility)

Plan the route to the destination.

Remember where every other car is and  
calculate where they will be in the next few  
seconds.

React to unforeseen issues like a child  
running in front of the car quickly.

High-level  
planning



Low-level  
planning

# Some Environment Types Revisited

**Fully observable:** The agent's sensors always show the whole **state**.

**vs.**

**Partially observable:** The agent only perceives part of the **state** and needs to remember or infer the rest.

**Deterministic:** **Percepts** are 100% reliable and changes in the environment is completely determined by the current **state** of the environment and the agent's **action**.

**vs.**

**Stochastic:**

- **Percepts** are unreliable (noise distribution, sensor failure probability, etc.). This is called a stochastic sensor model.
- The **transition function** is stochastic leading to transition probabilities and a Markov process.

**Known:** The agent knows the **transition function**.

**vs.**

**Unknown:** The agent needs to **learn the transition function** by trying actions.

We will spend the whole semester on discussing algorithms that can deal with environments that have different combinations of these three properties.

# Conclusion

Intelligent agents inspire the research areas of modern AI

**Search** for a goal  
(e.g., navigation).

**Optimize** functions  
(e.g., utility).

Stay within given  
**constraints**

(constraint satisfaction problem;  
e.g., reach the goal without  
running out of power)

Deal with **uncertainty**  
(e.g., current traffic on the  
road).

**Learn** a good agent  
program from data  
and improve over time  
(machine learning).

**Sensing**  
(e.g., natural language  
processing, vision)