Artificial Intelligence

Augustin Cosse.

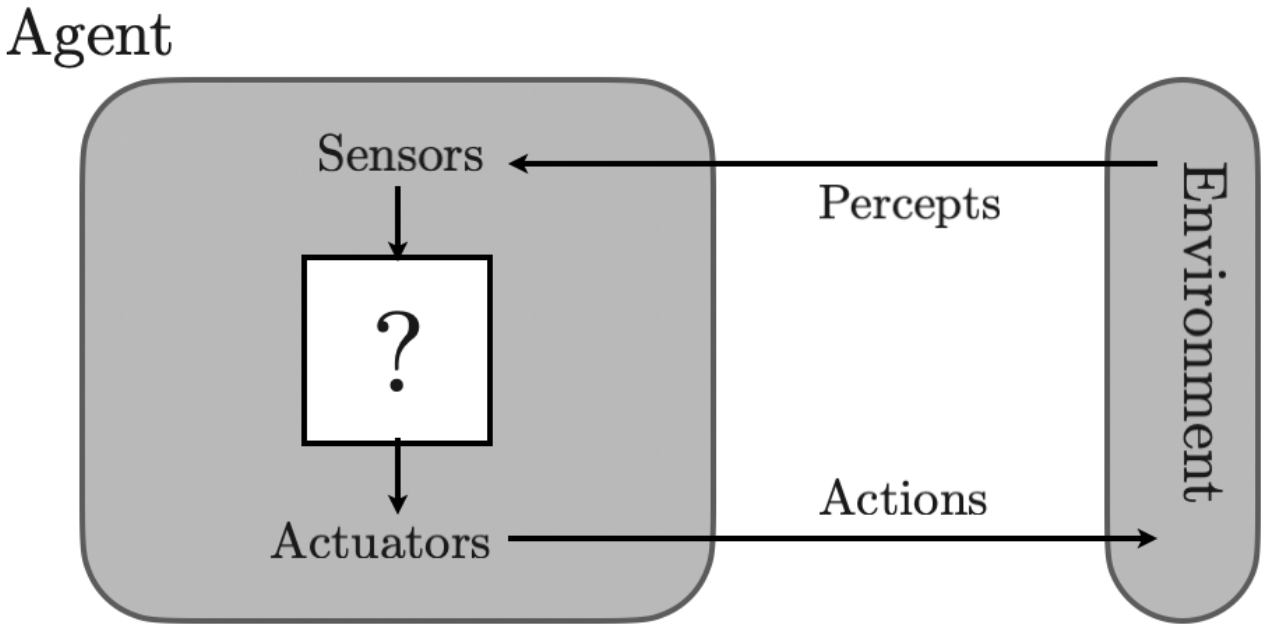
Fall 2020

September 18, 2021

Intelligent agents

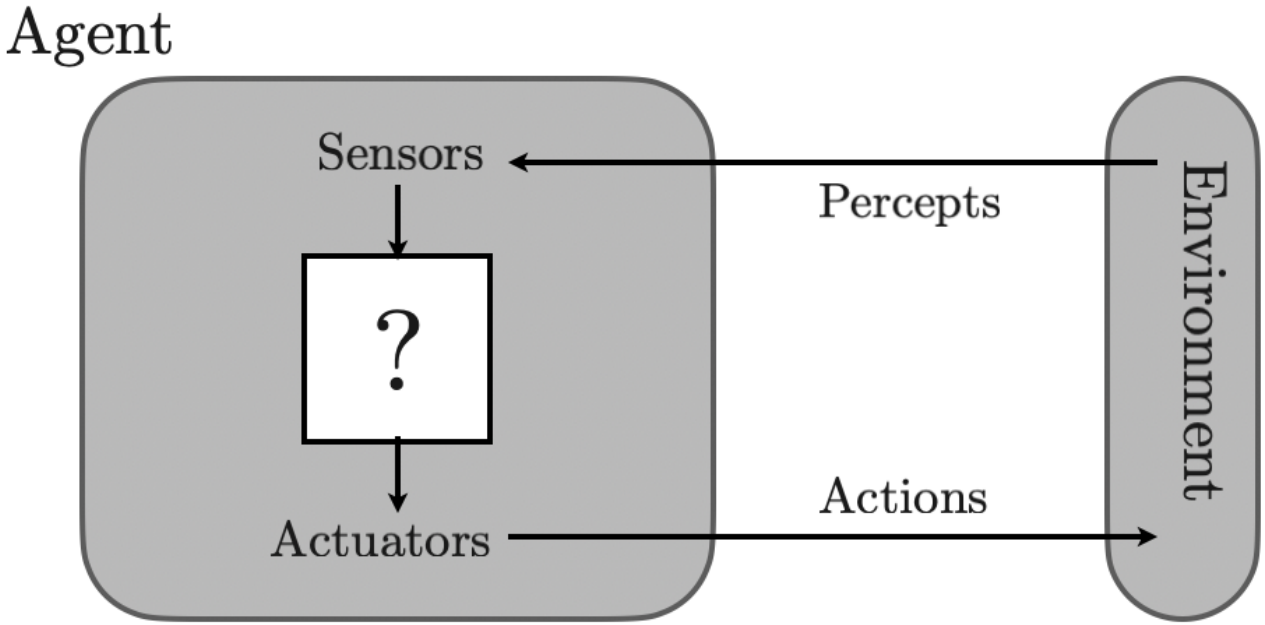
- Intelligent agent = Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

- Objective of the course = design agents that do a good job of acting on their environment



Intelligent agents

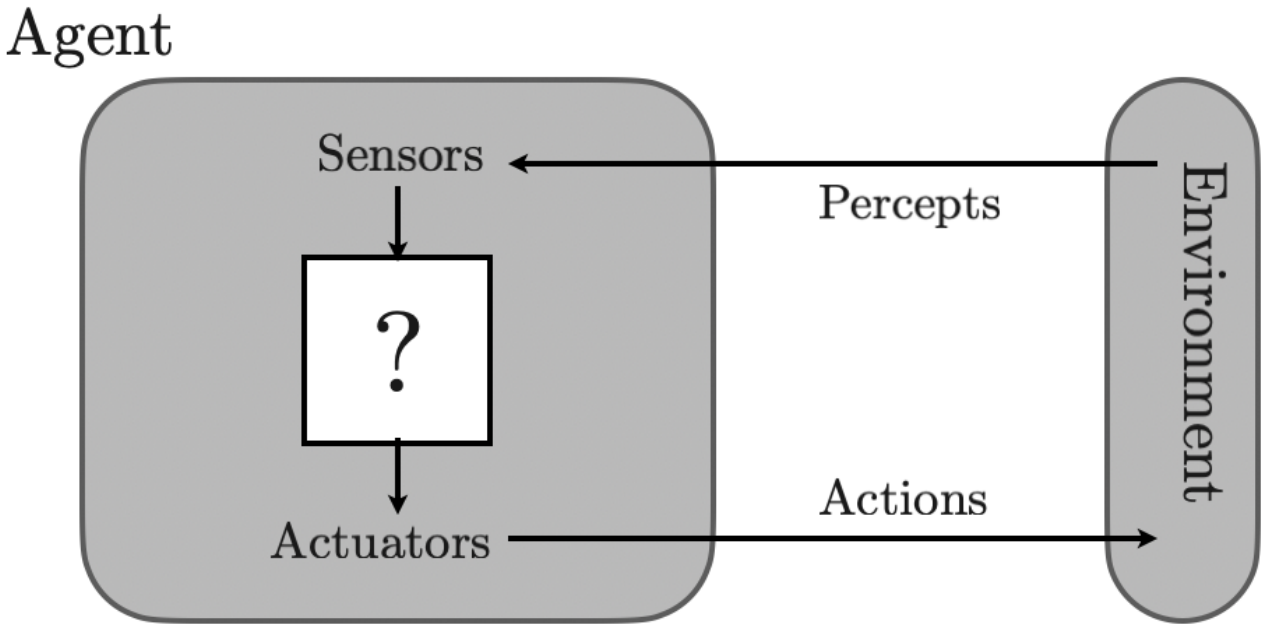
- We will use the term percept to refer to an agent's perceptual inputs at any given instant
- An agent percept sequence is the complete history of everything the agent has ever perceived



Intelligent agents

- Mathematically speaking, we say that an agent's behavior is described by the agent function which maps any given percept sequence to actions

- The agent function will be implemented by the agent's program

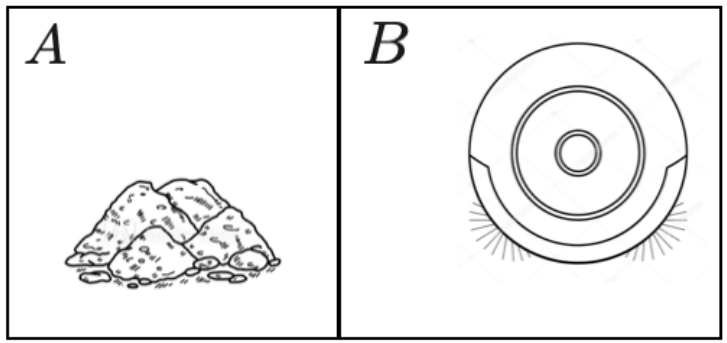


Rational agent

- When evaluating the performance of an agent, we will tend to rely on objective measures. E.g. for a vaccum cleaner, one performance measure could be the amount of dirt left on the floor. Another could be the electricity consumption.
- The decision of when to measure performance is also very important. In most applications, we will want to measure performance over the long run.
- 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 at the time.

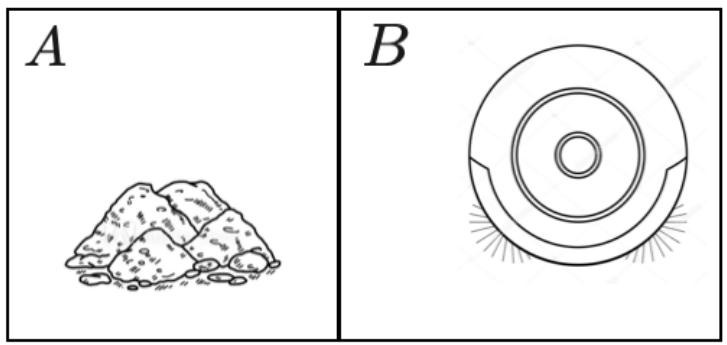
Rational agent

- As a general rule, we should design performance measures according to what we want in the environment and not how one thinks the agent should behave
- Consider the example of an autonomous vacuum cleaner (below). If we don't add penalties for the vaccum cleaner moves and if we program this vacuum cleaner to suck up the dirt when it reaches a cell where there is dirt and then move to the other cell, this agent can be considered rational.



Rational agent

- However, if we now consider the same agent (with a sequence of two actions: sucking up dirt then move) and define our performance measure to penalize extra moves, such a simple agent is not rational anymore as it will continue to move even when all the dirt is gone.



Rational agent

- Generally speaking, we should also not blame an agent for failing to take into account an event that it could not perceive.
- An example of this is an agent that would be facing a crosswalk and wanting to move across a street with no traffic. It might conclude that it is safe to cross, yet at the moment it takes its decision, a truck comes out of nowhere driving at full speed.
- We say that an agent is ominscient when it knows the exact outcome (no uncertainty) of its actions (including the full speed truck)

Rationality vs Omniscience

- Rationality is not the same as omniscience. Rationality maximizes expected performance while Perfection (omniscience) maximizes actual performance.
- The definition of rationality does not requires omniscience because the rationality of an action depends only on the sequence of percepts to date.
- Rationality of an action depends on four notions:
  - The performance measure
  - The percept sequence (all the information that the agent has perceived so far)
  - The agent prior knowledge on the environment
  - The actions that the agent can perform

Rationality vs Omniscience

- In short: 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 knowlege the agent has.

- An important aspect of rationality is information gathering (i.e. exploration). A rational agent should select actions that will diversify its percepts sequence and learn as much as possible from those percepts.

Autonomous agents

- By learning, we mean that the agent's initial configuration can reflect some prior knowledge but as the agent gains experience, this should be modified and augmented.
- If the agent's actions are based solely on built-in knowledge (i.e. prior knowledge from its designer), such that it doesn't need to pay no attention to its percepts, then we say that the agent lacks autonomy.
- A rational agent should be autonomous: it should learn what it can to compensate for partial or incorrect prior knowledge.
- One seldom requires autonomy from the start though. When the agent has little or no experience, it would have to act randomly unless the designer gave some assistance.

Intelligent agents

- Just as evolution provides animals with a sufficient number of built in reflexes to survive long enough to learn for themselves, it would be reasonable to provide an artificial agent with some initial knowledge as well as an ability to learn.
- After sufficient experience of its environment, the behavior of the agent can become independent of this initial knowledge.

Table Lookup Agent

- The simplest possible way to design an agent's program is to keep track of a Lookup Table.
- A **Table LookUp Agent** or **Table Driven Agent** operates by keeping in memory its entire percept sequence, and use a table to encode the mapping between the percept sequence and corresponding actions.

Agent Table-driven Agent(*percept*) **returns** action ;
persistent: percepts (a sequence initially empty) ;
table (a table of actions, indexed by
percept sequences, intially fully specified);

append percept to the end of percepts;
action $\leftarrow$ LOOKUP(*percepts*, *table*);
return *action*

Table Lookup Agent

- Keeping track of an exhaustive table is, in general, not needed. As an example, if we wanted to implement an agent that would compute the square root of any number, it would be wiser to require that square root to be accurate up to some precision. in this case, we can thus replace the table with a function returning Newton steps.

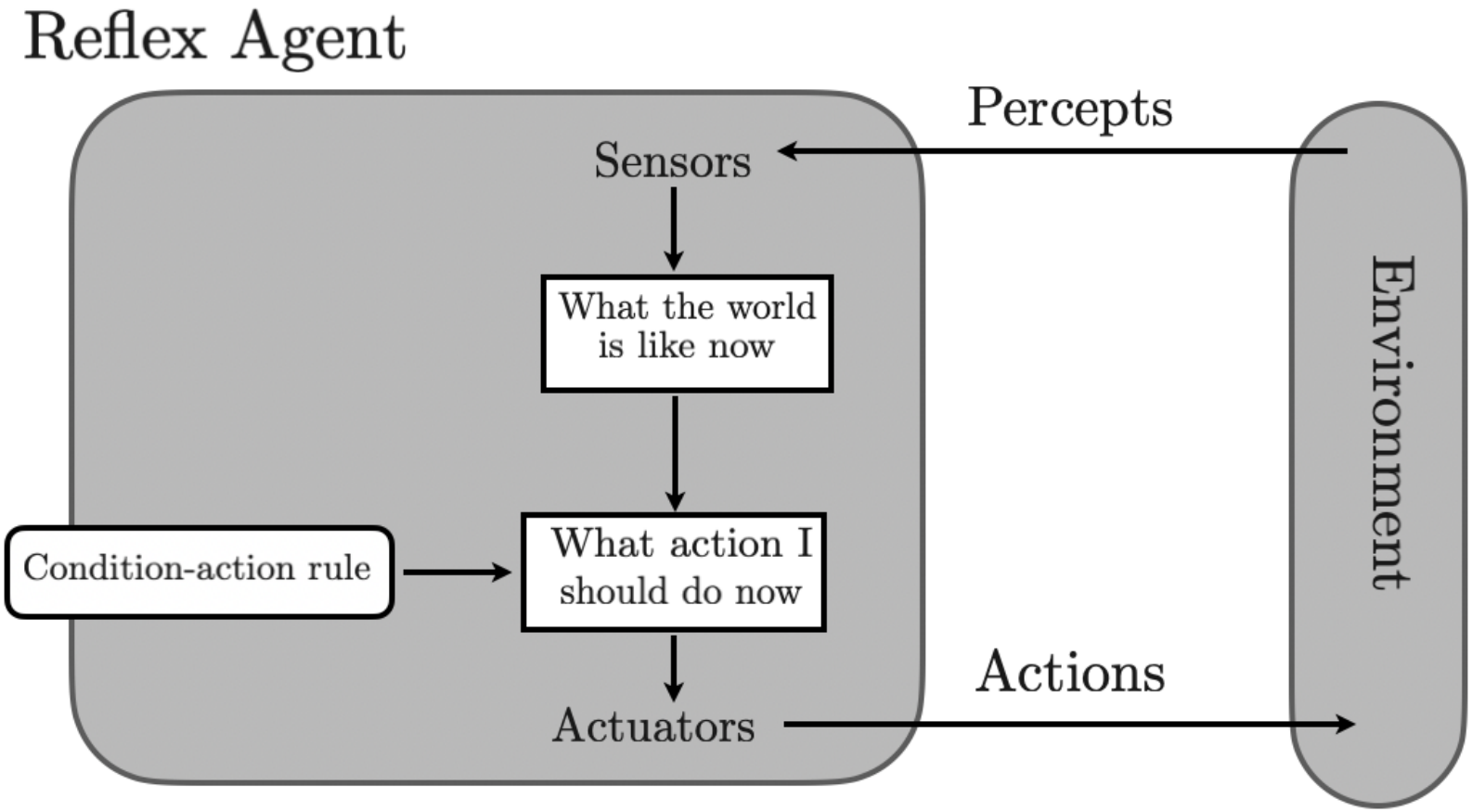
Agent Table-driven Agent(*percept*) **returns** action ;
persistent: percepts (a sequence initially empty) ;
table (a table of actions, indexed by
percept sequences, intially fully specified);

append percept to the end of percepts;
action $\leftarrow$ LOOKUP(*percepts*, *table*);
return *action*

Simple reflex Agents

- Although building a complete LOOKUP table is often too costly (if $|P|$ is the number of percepts and T is the lifespan of the agent, the table should contain $\sum_{t=1}^{T} |P|^t$ entries), we can sometimes summarize portions of the table by noting certain commonly occurring input/output associations.
- An example of this is an autonomous vehicle that would be behind another vehicle that brakes. We could imagine implementing a rule telling the autonomous vehicle that if the warning light of the car ahead comes on, then it should immediately brake.
- Such a rule is called condition-action rule and is the motivation for reflex based agents

Simple reflex Agents



Simple Reflex Agent

- The INTERPRET-INPUT function generates an abstracted description of the current state from the percept and RULE-MATCH returns the first rule in the set of rules that matches the given state description.

- Simple reflex agents are of limited intelligence. Such agents will work well only if the correct decision can be made on the basis of the current percept - that is only if the environment is fully observable

Agent Simple Reflex Agent(*percept*) **returns** 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

- Note that a reflex agent does not need to be rational.
- Consider the two cells vacuum cleaner. Imagine a reflex agent that receive as percept whether its current cell is dirty or not. The only decision the vacuum cleaner can make is whether to move into the next cell or not. If the agent is not told whether there is dirt on the adjacent cell, there is the risk for the agent to move to a cell where there is no dirt, or to stop moving and hence miss a cell where there is dirt
- On the contrary, given knowledge of the complete environment, this same reflex agent can be perfectly rational.

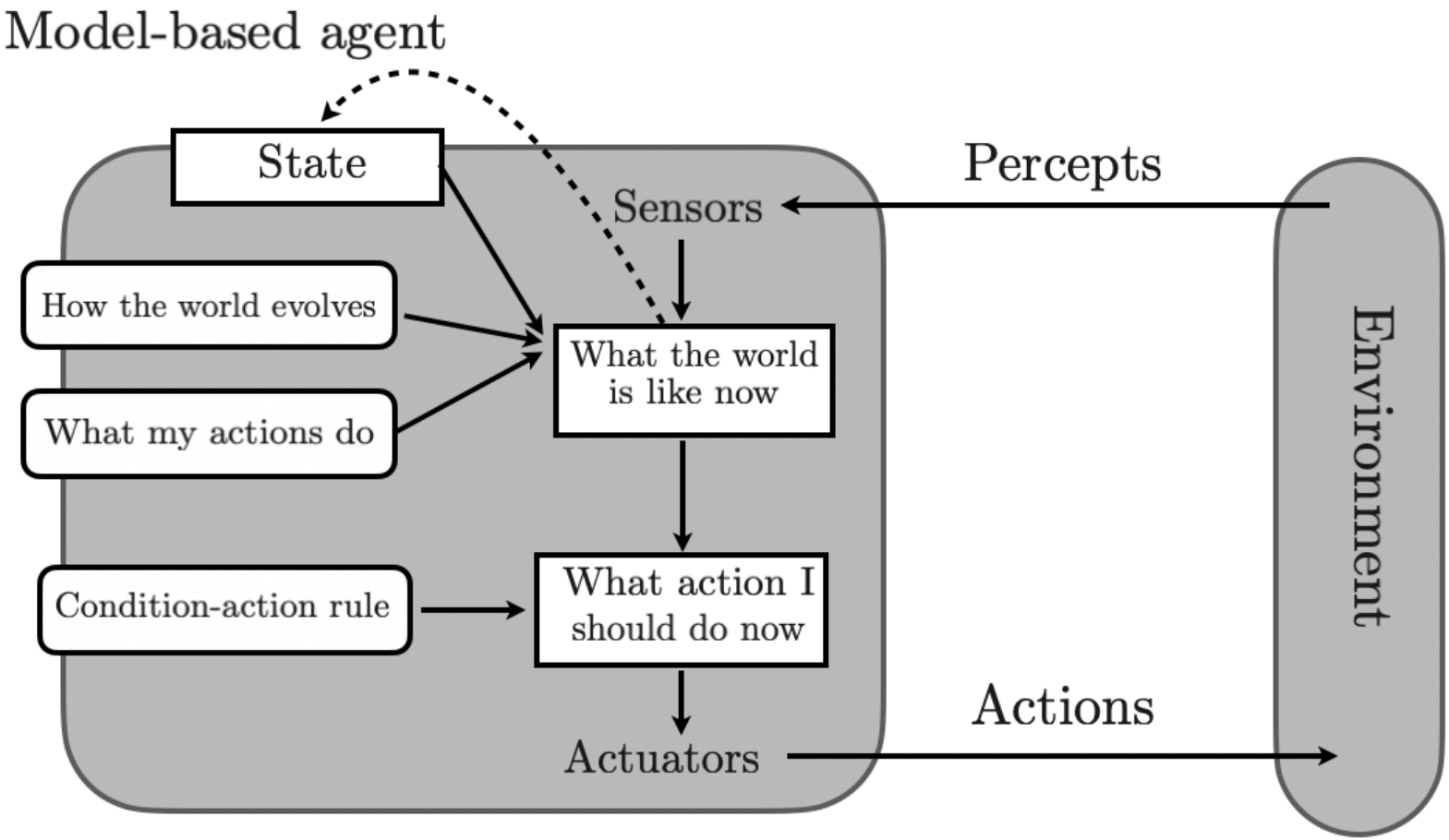
Keeping track of the world

- In several instances, only relying on the current percept might not be sufficient (think of some occlusion occuring right at the time where the brake ligth comes on for example)
- The problem illustrated by this example arises because the sensors do not provide access to the complete state of the world (we talk about partial observability). The most effective way to handle partial observability is for the agent to be able to predict the part of the world it cannot see now. That is to say the agent should maintain some sort of internal state that depends on the percepts history and can reflect some of the unobserved aspects of the current state.

Keeping track of the world

- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
  - First, we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago.
  - Second, we need some information about how the agent's own actions affect the world (e.g. when the agent turns the steering wheel clockwise, the car actually turns to the right)
- This knowledge about how the world evolves is called a model of the world. An agent that relies on such a model is called model based agent.

Model based agents



Model based Agent

- The interesting part in a Model Based Reflex Agent is the UPDATE-STATE function which is responsible for creating the new internal state description. The details of how models and states are represented vary widely depending on the type of environment and the particular technology used in the agent design.

Agent Model Based Reflex Agent(*percept*) **returns** action ;
persistent: *state* (the agent current conception of the world state) ;
model (how the next state depends on current state and action);
rules (a set of condition-action rules);
action (the most recent action, initially none);
state ← UPDATE-STATE(*percept*);
rule ← RULE-MATCH (*state, rules*);
action ← *rule*.ACTION;
return *action*

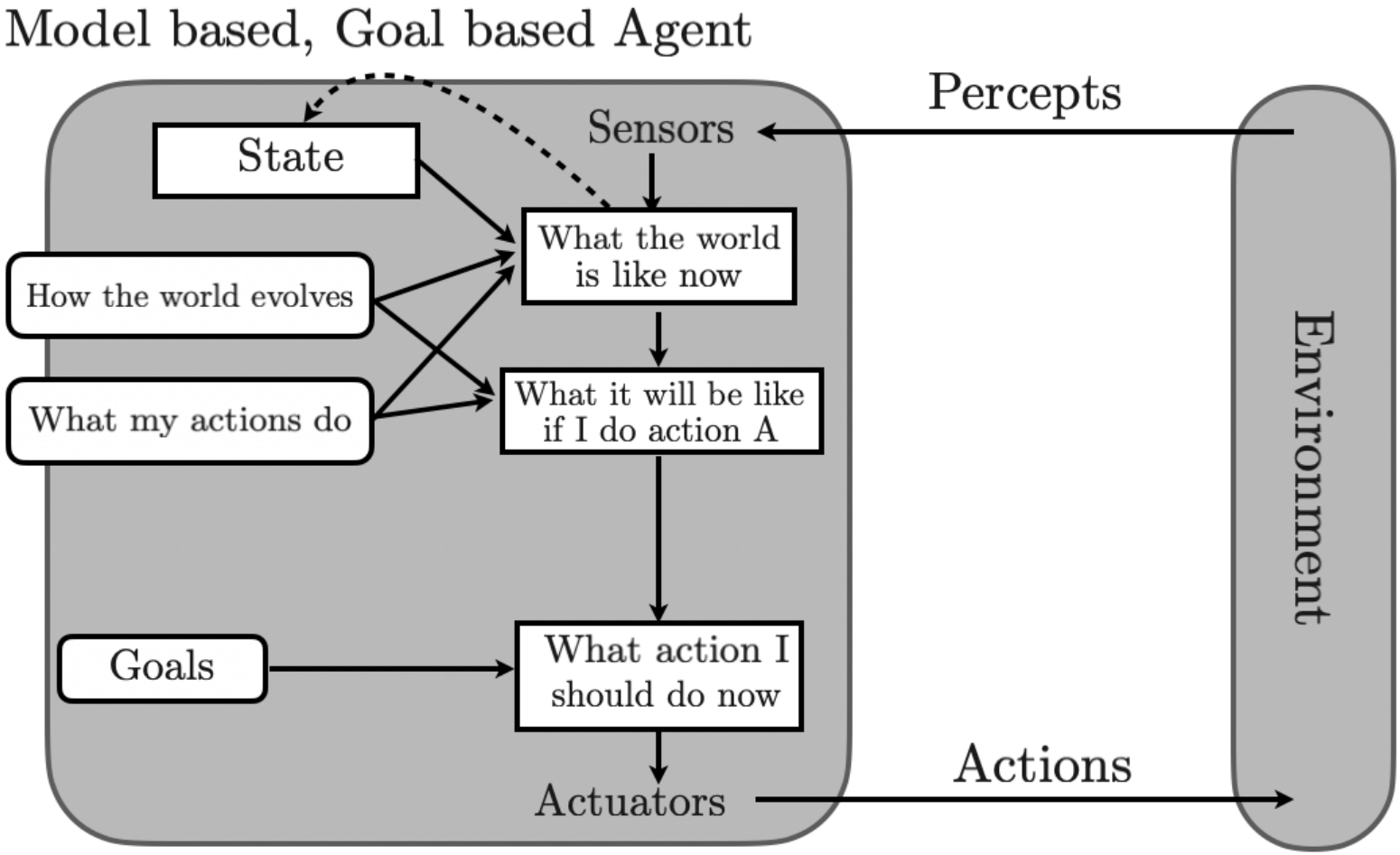
Goal based and Utility based agents

- Knowing about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, right, or go straight on. The right decision depends on where the taxi is trying to go.
- In other words, as well as a current state description, the agent needs some sort of goal information, which describes situations that are desirable
- The agent program can combine this with the model to choose actions that achieve the goal.

Goal based and Utility based agents

- Sometimes goal-based action selection is straightforward (e.g. when goal satisfaction immediately results from a single action) sometimes it will be more tricky (for example when the agent has to consider a long sequence of twists and turns in order to achieve the goal).
- Search and Planning are the subfields of AI devoted to finding action sequences that achieve the agents's goal.
- Note that this is fundamentally different from the reflex agent approach as it involves consideration of the future (e.g. "What will happen if I do this and this?", or "will that make me happy?") In the reflex based agent this information is not present as the built-in rule is a direct map between percepts and actions.

Goal based agents



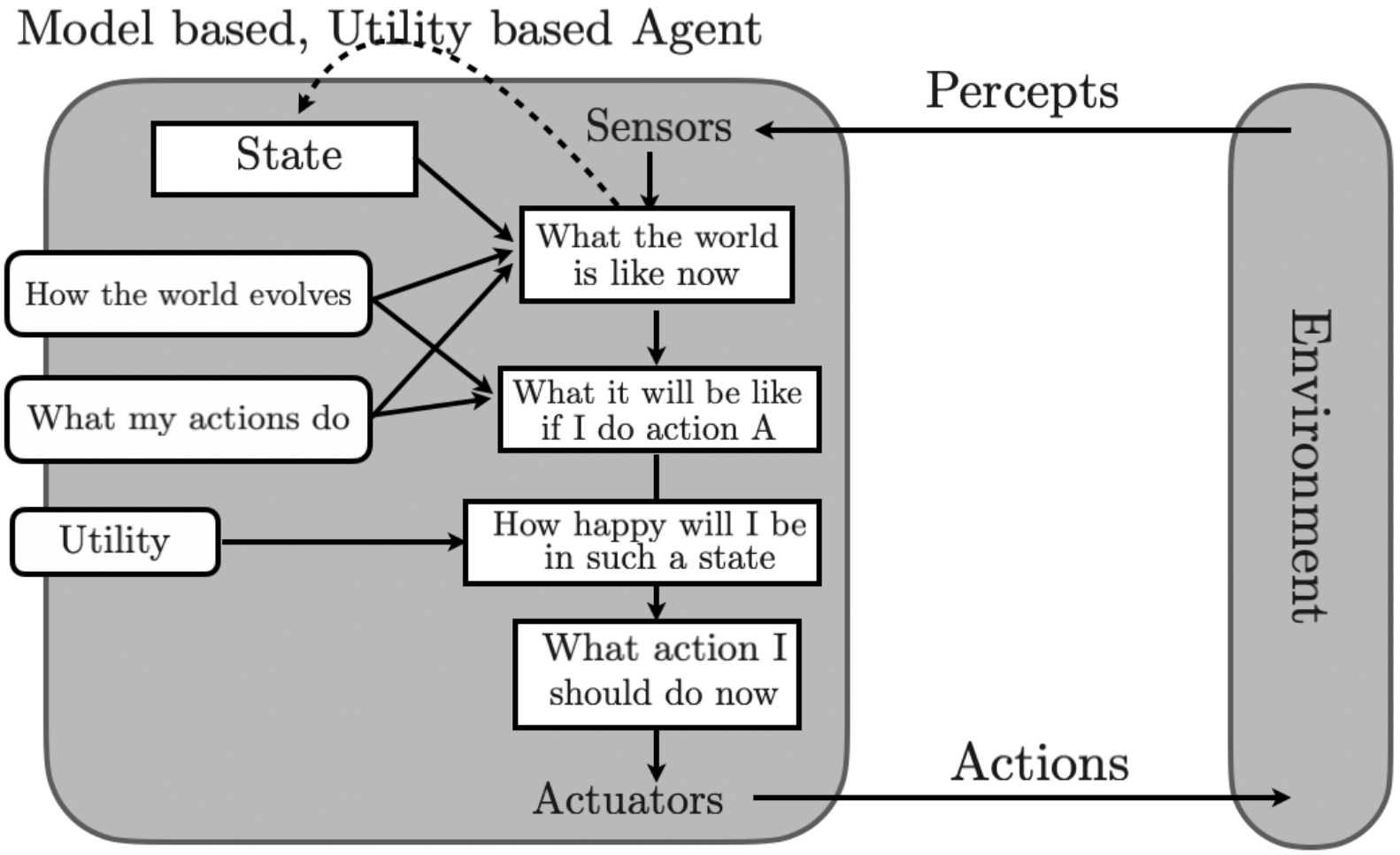
Goal based and Utility based agents

- Goals alone are not really enough to generate high-quality behavior. For example, there are many action sequences that will get the taxi to its destination, thereby achieving the goal, but some are quicker, safer, more reliable, or cheaper than others
- Goals just provide a crude distinction between "happy" and "unhappy" states yet a more general performance measure should allow a comparison of different world states (or sequences of states) according to exactly how happy they would make the agent if they could be achieved
- The customary terminology is to say that if one world state is preferred to another, then it has higher utility for the agent

Goal based and Utility based agents

- Utility is the function that maps a state onto a real number describing the associated degree of happiness
- The utility function allows rational decisions to be made by the agent in two situations where goals can reveal inadequate:
  - First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate trade-off
  - Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed up against the importance of the goals

Goal based agents



Learning agent

- So far we have described agents with various methods for selecting actions. We haven't explained how the agent programs come into being.
- In his seminal paper, Turing (1950) considers the idea of programming his intelligent machine by hand. He estimates how much work this might take and concludes that "Some more expeditious methods seems desirable"
- The method Turing proposes is to build learning machines and then teach them. In many areas of AI, it is now the preferred method for creating state-of-the-art systems.
- Learning has another advantage: as we noted earlier, it allows the agent to operate in initially unknown environments and to become more and more competent compared to what the initial knowledge alone might have allowed.

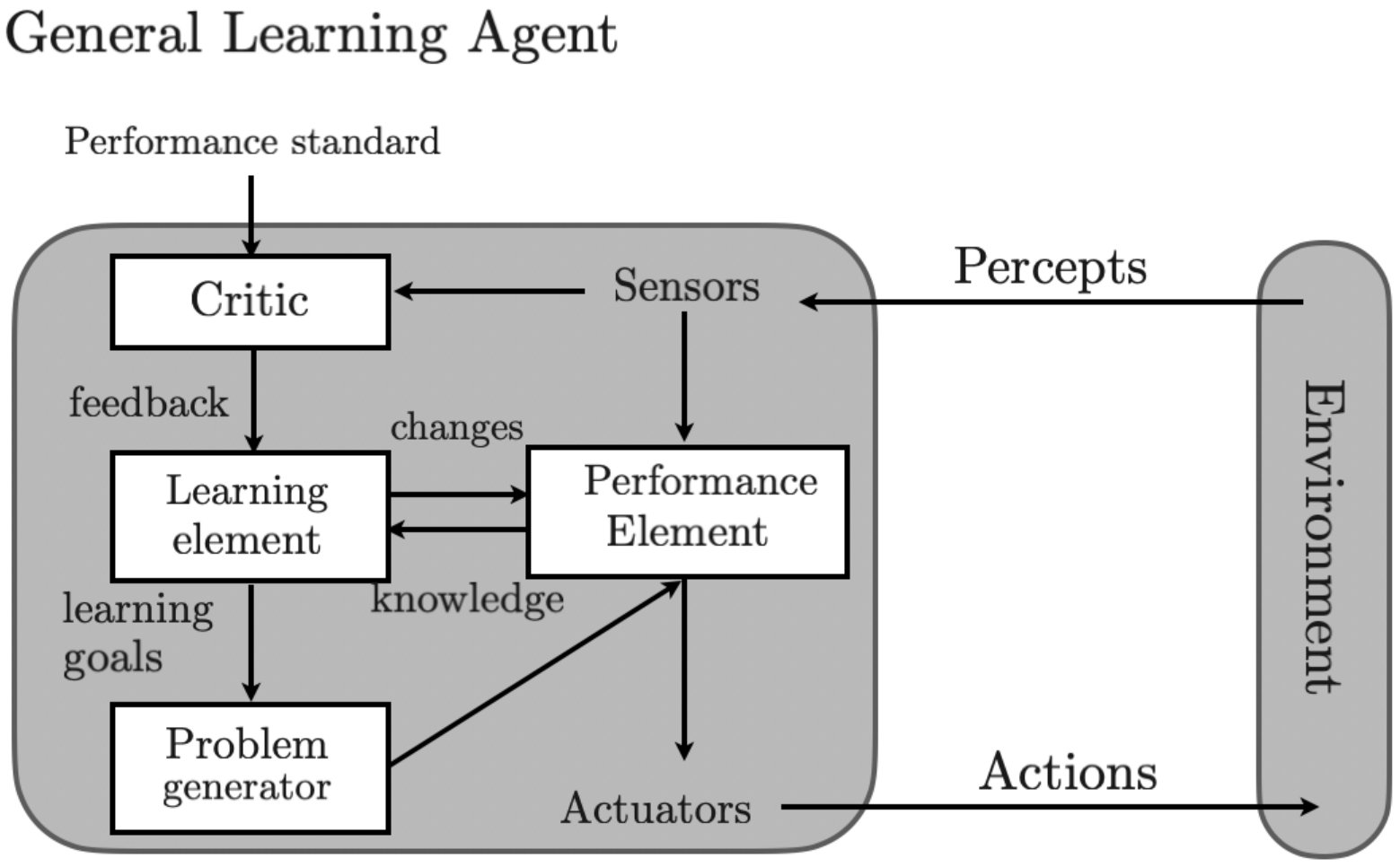
Learning agent

- A learning agent relies on 4 conceptual components:
- The most important elements are the learning element (responsible for making improvements) and the performance element (responsible for selecting external actions).
- The performance element is what we have previosuly considered to be the complete agent (it takes in percepts and decides on actions)
- The learning element receives 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

Learning agent

- The critic tells the learning element how well the agent is doing with respect to a fixed performance standard
- The critic is needed because the percepts themselves provide no indication of the agent's success. I.e. a chess program could receive a percept indicating that it has checkmated its opponent but it will need a performance standard to know if this is a good thing.
- The last component of the learning agent is the problem generator which is responsible for suggesting actions that will lead to new and informative experiences
- The point is that if the performance element had the final word, it would keep doing the same actions that are best, given what it knows. But if the agent is willing to explore (and do perhaps some slightly suboptimal actions in the short term), it might discover much better actions in the long run.

Goal based agents



The (task) environment

- A (task) environment is essentially the problem to which the rational agent is the solution.
- Under the name task environment, we group the performance measure, the environment, the agent's actuators and the agent's sensors.

Agent Type	**Performance measure**	**Environment**	**Actuators**	**Sensors**
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, diagnoses, treatments, ..	Keyboard entry of symptoms, findings, patient symptoms,..
Satellite image analysis system	Correct image categorization	telecom link from orbiting satellite	Display of scene categorization	Color pixel array

The (task) environment

- Fully observable vs partially observable. If the agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.
- An environment might be partially observable because of noisy or inaccurate sensors or because parts of the state are just missing from the sensor data (e.g. a vacuum cleaner with only local sensors cannot tell whether there is dirt in other rooms)
- If the agent has no sensors, then the environment is unobservable

The (task) environment

- A taks environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action
- Fully observable environments are convenient because the agent does not need to maintain any internal state to keep track of the world.

The (task) environment

- Single agent vs Multiagent. An agent solving a crossword puzzle by itself is a single agent environment while an agent playing chess is in a two-agent environment.
- Deterministic vs Nondeterministic. An environment is called deterministic when its next state is completely determined by its current state and the action taken by the agent. An inaccessible (or too complex) environment may appear nondeterministic.
- Episodic vs Nonepisodic. In an episodic environment, the agent's experience is divided into episodes. Each episode consists of the agent perceiving and then acting. The quality of each action only depends on what was learned during the episode.

The (task) environment

- Static vs Dynamic. If the environment can change while the agent is deliberating, we say that the environment is dynamic for the agent. Static environments are easier as they do not require the agent to constantly keep looking at the world.
- Discrete vs Continuous. When there is a limited number of clearly defined percepts and actions, we say that the environment is discrete. Chess is discrete, taxi driving is continuous.