Artificial Intelligence

Augustin Cosse.

Fall 2020

September 9, 2020

Intelligent agents

- Anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors
- Objective of the course = design agents that do a good job of acting on their environment

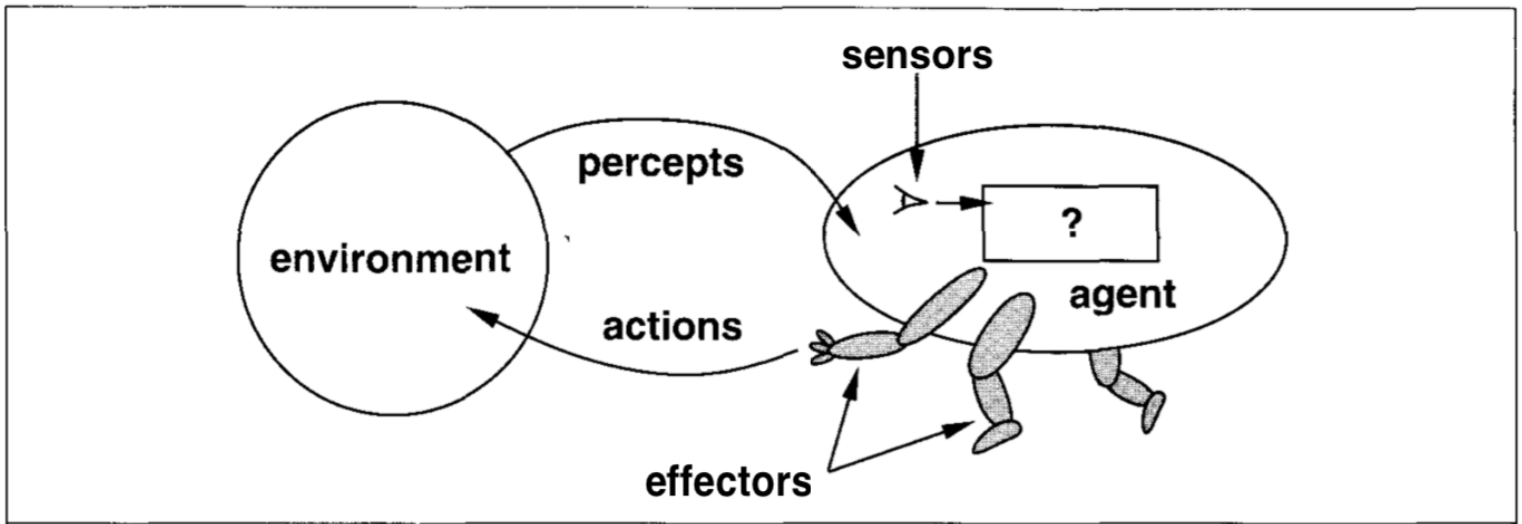

Figure 2.1 Agents interact with environments through sensors and effectors.

Intelligent agents

- 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.
- 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 on some sidewalk, on a street without too much traffic and would someone with whom it has to talk on the other side of the street. It might conclude it is safe to cross while all of a sudden, a truck comes out of nowhere riding at full speed.

Intelligent agents

- We say that an agent is ominscient when it knows the exact outcome of its actions (including the full speed truck)
- Rationality on the other hand is concerned with expected success given what has been perceived (crossing the street at that point is rational although it can lead to a failure)

Intelligent agents

- Rationality of an action depends on four points:
  - Performance measure
  - All the information that the agent has perceived so far. We will call this the percept sequence
  - What the agent knows on the environment (percepts + past learning)
  - The actions that the agent can perform

Intelligent agents

- Once we realize that an agent's behavior depends only on its percept sequence to date, then we can describe any particular agent by making a table of the action it takes in response to each possible percept sequence
- For most agents, this would be a very long listinfinite, in fact, unless we place a bound on the length of percept sequences we want to consider
- Such a list is called a mapping from percept sequences to actions
- A good approach is to let the agent behave a random while we build such a mapping and keep track of the average score of the agent.

Intelligent agents

- Specifying which action an agent ought to take in response to any given percept sequence provides a designfor an ideal agent. In general, however, such an approach is intractable.
- In general keeping track of an exhaustive table is 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.

Intelligent agents

- If the agent's actions are based completely on built-in knowledge, such that it need pay no attention to its percepts, then we say that the agent lacks autonomy
- It would be too stringent, though, to require complete autonomy from the word go: when the agent has had little or no experience, it would have to act randomly unless the designer gave some assistance.
- Just as evolution provides animals with enough built-in reflexes so that they can survive long enough to learn for themselves, it would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn.

Intelligent agents

- All the agent programs that we will consider in this course will have the same structure: accepting percepts from an environment and generating actions
- The early versions of agent programs will have a very simple form. Each will use some internal data structures that will be updated as new percepts arrive

```
function SKELETON-AGENT(percept) returns action
  static: memory, the agent's memory of the world

  memory ← UPDATE-MEMORY(memory, percept)
  action ← CHOOSE-BEST-ACTION(memory)
  memory ← UPDATE-MEMORY(memory, action)
  return action
```

Figure 2.4 A skeleton agent. On each invocation, the agent's memory is updated to reflect the new percept, the best action is chosen, and the fact that the action was taken is also stored in memory. The memory persists from one invocation to the next.

Intelligent agents

- The agent program receives only a single percept as its input. It is up to the agent to build up the percept sequence in memory.
- The goal or performance measure is not part of the skeleton program. This is because the performance measure is applied externally to judge the behavior of the agent

Table Lookup Agent

- The simplest possible way we can think of to write the agent 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 such percept sequences and corresponding actions.

```
function TABLE-DRIVEN-AGENT(percept) returns action
  static: percepts, a sequence, initially empty
          table, a table, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Figure 2.5 An agent based on a prespecified lookup table. It keeps track of the percept sequence and just looks up the best action.

Simple reflex Agents

- Although building a complete LookUp table is often too costly, 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 brake lights of the car in front come on, then it should initiate braking
- Such a rule is called condition-action rule

Simple reflex Agents

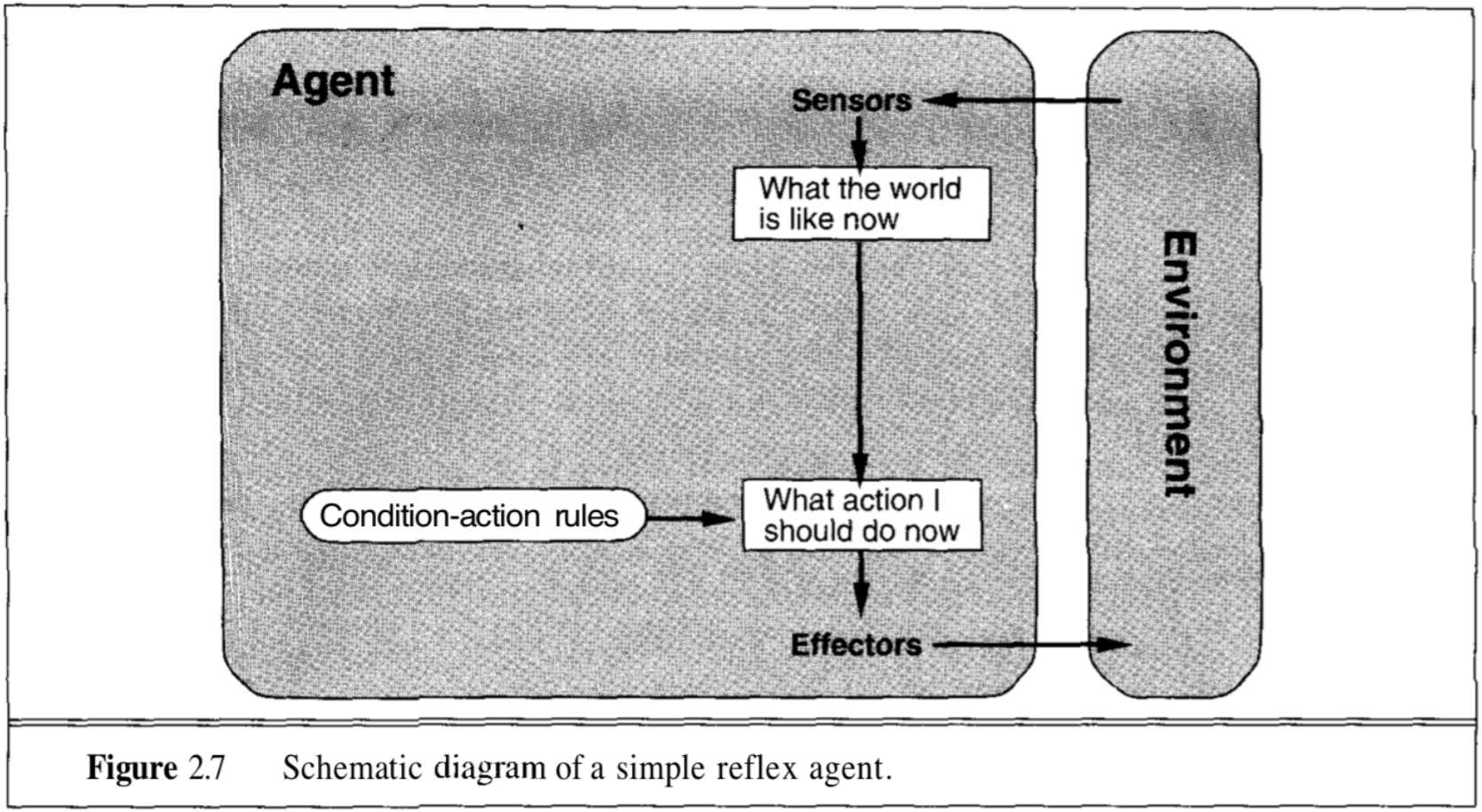

Figure 2.7 Schematic diagram of a simple reflex agent.

Simple reflex Agents

```
function SIMPLE-REFLEX-AGENT(percept) returns action
  static: rules, a set of condition-action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← RULE-ACTION[rule]
  return action
```

Figure 2.8 A simple reflex agent. It works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

Keeping track of the world

- The simple reflex agent described before will work only if the correct decision can be made on the basis of the current percept. In several instances through, only relying on the current percept might not be sufficient (think of some occlusion occuring right at the time where the brake ligth turns on for example)
- The problem illustrated by this example arises because the sensors do not provide access to the complete state of the world. In such cases, the agent may need to maintain some internal state information

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 agentfor 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

Utility based agents

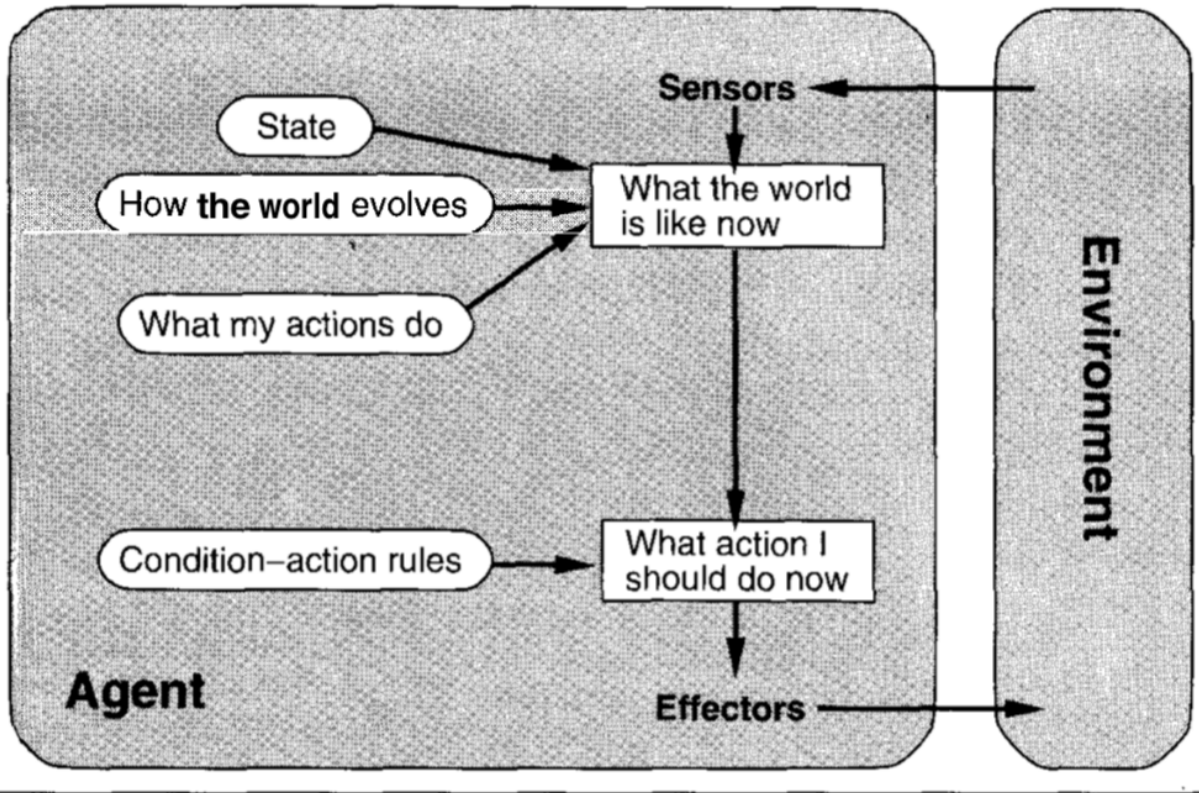

Figure 2.9 A reflex agent with internal state.

Goal based and Utility based agents

- Knowing about the current state of the environment is not always enough to decide what to do. j For example, at a road junction, the taxi can turn left, right, or go straight on. The right decision j depends on where the taxi is trying to get to
- 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 information about the results of possible actions in order to choose actions that achieve the goal.
- Sometimes j this will be simple, when goal satisfaction results immediately from a single action; sometimes, j it will be more tricky, when the agent has to consider long sequences of actions (cf Search and planning)

Goal based agents

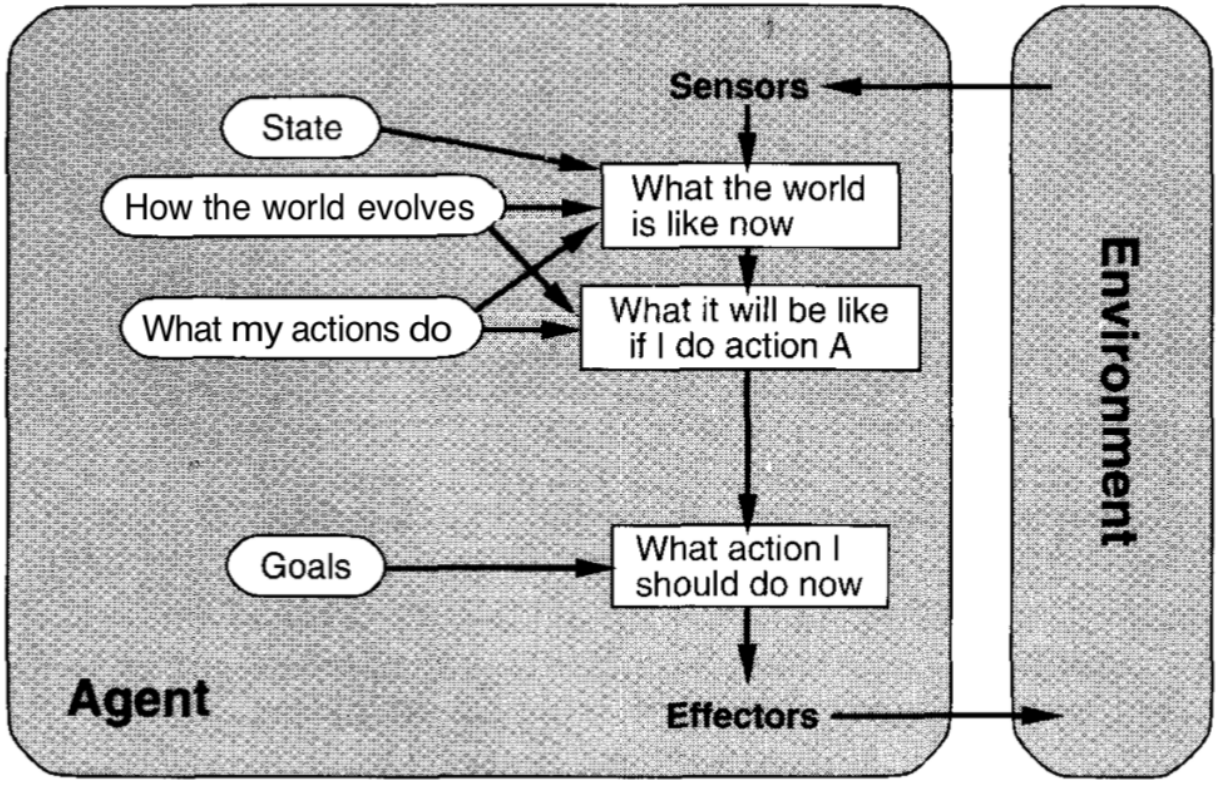

Figure 2.11 An agent with explicit goals.

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 I 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 therefore a function that maps a state9 onto a real number, which describes the j associated degree of happiness
- Utility function allows rational decisions in two kinds of cases where goals have trouble:
  - First, when there are conflicting goals, 1 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

The environment

- Accessible vs Inaccessible. An environement is effectively accessible if the sensor can detect all aspects that are relevant to the choice of action
- 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 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 environment 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.