## 2. Intelligent Agents

## 2.1 Agents and environment

An **agent** is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. We use the term **percept** to refer to the agent’s perceptual inputs at any given instant. An agent’s **percept sequence** is the complete history of everything the agent has ever perceived

In our *reinforcement learning algorithm*, the agent is the instructor or core of the system. The agent is equipped with ”sensors” that enables it to receive observations and rewards from the environment, before returning a new action based on that information. In order for an agent to learn, it is dependant on two components; a *policy* and a *learning method*.

The policy denoted by π, is a function that maps an observation to an action, π(a|s). We want our agent to attempt to learn a policy that can maximise the expected sum of rewards by choosing the best action given an observation. We use a learning method to update the policy’s parameters in order to find this better action.

The agent lives and operates in an environment. As mentioned earlier, the agent receives observations from the environment. This can be thought of as the way the environment interacts back with the agent. We also define the state as the precise location and time in which the agent finds itself in the environment.

A graphical representation of a the cyclic interaction between the agent and the environment. The agent executes an action at given some state st and reward rt in the environment. The environment then outputs the next state st+1 and reward rt+1, which is used as feedback to the agent

Diagram, schematic

Description automatically generated

**How can we define if an agent’s behaviour is good or bad?**

The right action should cause the agent to be most successful

When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well. This notion of desirability is captured by a performance measure that evaluates any given sequence of environment states

*As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.*

### 2.2 Rationality

What is rational at any given time depends on four things:

* The performance measure that defines the criterion of success.
* The agent’s prior knowledge of the environment.
* The actions that the agent can perform.
* The agent’s percept sequence to date

### 2.2.1 Omniscience, learning, and autonomy

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

**Information gathering**: Doing actions in order to modify future precepts. Another example of information gathering is provided by the exploration that must be undertaken by an agent in an initially unknown environment. Our definition requires a rational agent not only to gather information but also to **learn** as much as possible from what it perceives

To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, 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. After sufficient experience of its environment, the behavior of a rational agent can become effectively independent of its prior knowledge.

### 2.3 Specifying the task environment

**Task environment**. PEAS (Performance, Environment, Actuators, Sensors) description

Table

Description automatically generated

**In depth example of a task environment**

Table

Description automatically generated

*Performance measures* of the previously introduced task environment: Desirable qualities include getting to the correct destination; minimizing fuel consumption and wear and tear; minimizing the trip time or cost; minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort; maximizing profits.

The driving *environment*. Any taxi driver must deal with a variety of roads, ranging from rural lanes and urban alleys to 12-lane freeways. The roads contain other traffic, pedestrians, stray animals, road works, police cars, puddles, potholes etc. The taxi must also interact with potential and actual passengers. There are also some optional choices. The taxi might need to operate in Southern California, where snow is seldom a problem, or in Alaska, where it seldom is not. It could always be driving on the right, or we might want it to be flexible enough to drive on the left when in Britain or Japan. Obviously, the more restricted the environment, the easier the design problem

The *actuators* for an automated taxi include those available to a human driver: control over the engine through the accelerator and control over steering and braking. In addition, it will need the output of a display screen or voice synthesizer to talk back to the passengers, and perhaps some way to communicate with other vehicles, politely or otherwise.

The basic *sensors* for the taxi will include one or more controllable video cameras so that it can see the road; it might augment these with infrared or sonar sensors to detect distances to other cars and obstacles. To avoid speeding tickets, the taxi should have a speedometer, and to control the vehicle properly, especially on curves, it should have an accelerometer. To determine the mechanical state of the vehicle, it will need the usual array of engine, fuel, and electrical system sensors. Like many human drivers, it might want a global positioning system (GPS) so that it doesn’t get lost. Finally, it will need a keyboard or microphone for the passenger to request a destination

**Fully observable** vs. **partially observable**: If an 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

For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A’s performance measure. Thus, chess is a **competitive** **multiagent** environment.

In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** **multiagent** environment.

If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is **deterministic**; otherwise, it is **stochastic.**

We say an environment is **uncertain** if it is not fully observable or not deterministic

**Episodic, Sequential environments**

**Static** vs. **dynamic**: If the environment can change while an agent is deliberating, then we say the environment is **dynamic** for that agent; otherwise, it is **static**

## 2.4 The structure of agents

### 2.4.1 Agent programs

**Agent program:** The agent program is a concrete implementation, running within some physical system

**Agent function:** The agent function is an abstract mathematical description

Graphical user interface, text

Description automatically generated