# Agents And Environments

An **AGENT** is an ENTITY which perceives its environment through its SENSOR and acts upon that environment through its ACTUATORS.

A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators. A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators. A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.

A screenshot of a cell phone

Description automatically generated

The **PERCEPT** of an Agent is the set of all inputs that an agent perceives at a given instant.

The **PERCEPT SEQUENCE** is the sequence of all percepts that the agent has perceived so far.

The agent’s choice of action at any given time depends on its PERCEPT SEQUENCE but it cannot depend on what it has not perceived.

Mathematically speaking, it is possible to teach an agent what actions to execute by providing a function which is called **AGENT FUNCTION** that maps every possible percept sequence to an action.

Naturally, such a function could not be physically implemented because its DOMAIN would be HUGE or even INFINITE.

## Rational Agent

A Rational Agent is an agent that does the right thing according to what it knows or can do.

The concept of right thing can be established through a PERFORMANCE MEASURE which evaluates how desirable the actions of the agent have been. Every action of an agent results in a change of the environment and so the PERFORMANCE MEASURE evaluates how desirable the sequence of different environment states have been.

Therefore, the PERFORMANCE MEASURE evaluates the sequence of environment states and that is what we care about because the purpose of rational agents is to change the state of an environment in order to achieve a goal.

The PERFORMANCE MEASURE depends on the goal to be achieved and establishing it is not an easy task.

Consider, for example, the vacuum-cleaner agent from the preceding section. We might propose to measure performance by the amount of dirt cleaned up in a single eight-hour shift. With a rational agent, of course, what you ask for is what you get. 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, and so on. A more suitable performance measure would reward the agent for having a clean floor. For example, one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated). 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. Even when the obvious pitfalls are avoided, there remain some knotty issues to untangle. For example, the notion of “clean floor” in the preceding paragraph is based on average cleanliness over time. Yet the same average cleanliness can be achieved by two different agents, one of which does a mediocre job all the time while the other cleans energetically but takes long breaks. Which is preferable might seem to be a fine point of janitorial science, but in fact it is a deep philosophical question with far-reaching implications. Which is better— a reckless life of highs and lows, or a safe but humdrum existence? Which is better—an economy where everyone lives in moderate poverty, or one in which some live in plenty while others are very poor? We leave these questions as an exercise for the diligent reader.

Therefore, what is important to notice is that when establishing a PERFORMANCE MEASURE we have to establish it by thinking how we want the environment to be and we don’t have to focus on how the agent aims to achieve it.

For example, the duty of Janitors at school is to clean class rooms. A good performance measure is the average of clean class rooms in a day (it is an environment state we aim to achieve), the higher the latter is the better it is. We don’t have to care how that is achieved, it is a problem of Janitors.

It is possible to go even further and build a more complex definition of a rational agent.

The RATIONALITY of an agent depends on the following factors:

1. Its performance measure
2. Its percept sequence to date
3. Its range of actions (what it can do)
4. Its knowledge of the environment (what it knows)

From these 4 factors we can elaborate the final definition of a rational agent.

An agent is RATIONAL when for all possible percept sequences it chooses to perform an action which maximises its performance measure according to its capabilities (range of actions) and its knowledge of the environment.

## Omniscience vs Rationality

While a RATIONAL Agent maximises its **EXPECTED** performance, an OMNISCIENT Agent maximises its ACTUAL performance.

In other words, while a rational agent knows the expected outcome of an action, an omniscient agent knows the actual outcome of an action. An omniscient agent is like an agent that can see the FUTURE.

Naturally, OMNISCIENE is impossible in reality and agents are not expected to be OMNISCIENT.

Consider the following example: I am walking along the Champs Elys´ees one day and I see an old friend across the street. There is no traffic nearby and I’m not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner,2 and before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read “Idiot attempts to cross street.”

## How can rational agents maximise their performance measure as much as possible

In order to maximise their performance as much as possible , rational agents should be able to perform the following processes:

1. Information gathering
2. Learning

Information gathering consists of acquiring as much as possible information from the surrounding environment and rational agents can achieve this by performing actions. In other words, by doing actions rational agents can modify future percepts. For example, if an agent does not look both ways before crossing a busy road, then its percept sequence will not tell it that there is a large truck approaching at high speed. If the agent looked both ways (performed actions) then it would have known there was a truck arriving (would have modified its future percepts).

A rational agent can maximise its performance measure only if enough information is gathered from the surrounding environment.

If a rational agent can’t learn then it lucks autonomy. An agent that is not able to learn from the gathered information cannot behave rationally in those situations where its prior knowledge is incorrect because some exceptions have occurred. An agent that can learn can become independent from its prior knowledge by augmenting it. For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

The female sphex will dig a burrow, go out and sting a caterpillar and drag it to the burrow, enter the burrow again to check all is well, drag the caterpillar inside, and lay its eggs. The caterpillar serves as a food source when the eggs hatch. So far so good, but if an entomologist moves the caterpillar a few inches away while the sphex is doing the check, it will revert to the “drag” step of its plan and will continue the plan without modification, even after dozens of caterpillar-moving interventions. The sphex is unable to learn that its innate plan is failing, and thus will not change it.

From this last example we can see that the sphex does not behave as a rational agent because she does not augment its prior knowledge.

So now we may pose a question. Are humans rational agents?

Maybe the answer to the above question is that every one of us is rational even though we all behave differently because we all different PERFORMANCE MEASURES.

## Task Environment

Before designing an agent, it is best practice to identify the TASK ENVIRONMENT which the agent will be the SOLUTION.

The TASK ENVIRONMENT is also called the PEAS description (Performance, Environment, Actuators, Sensors).

The TASK ENVIRONMENT consists of the following elements:

1. PERFORMANCE MEASURE
2. ENVIRONMENT
3. ACTUATORS
4. SENSORS

A screenshot of a cell phone

Description automatically generated

It may come as a surprise to some readers that our list of agent types includes some programs that operate in the entirely artificial environment defined by keyboard input and character output on a screen. “Surely,” one might say, “this is not a real environment, is it?” In fact, what matters is not the distinction between “real” and “artificial” environments, but the complexity of the relationship among the behaviour of the agent, the percept sequence generated by the environment, and the performance measure.

In summary, what matters is the relationship about the agent and the environment and not whether or not the environment is real.

A screenshot of a cell phone

Description automatically generated

## Different kinds of Environments

According to the type of the environment, some agents are more suitable than others. This is why it is important to identify the task environment before-hand.

We can categorize a given task environment according to the type of environment which it embeds.

* **Fully observable environment =>**  an environment is fully observable if an agent through its sensors can detect all aspects of the environment which are relevant to its choice of actions.
* **Partially observable environment** => when an agent cannot detect all aspects of the environment that are relevant to its choice of action.
* **Unobservable environment** => when an agent has no sensors
* **Single-Agent environment** => when there is only one agent
* **Multi-agent environment** => when there is more than one agent
  + **Competitive environment** => when an agent maximises its performance measure it affects the performance measure of another agent by decreasing it
  + **Cooperative environment** => when agents working together maximise their performance measure
* **Deterministic environment** => when the next state of the environment is only determined by the current state and the action of the agent. A partially observable and deterministic environment may seem to be a stochastic one for the agent because it does not know all necessary information about the environment.
* **Stochastic environment** => when the next state of the environment does not depend on the current state and the action of the agent. There may be several possible states each of them having a different likelihood of occurring.
* **Non**-**deterministic environment** => similar to the stochastic environment but the possible next states are not marched to any probability of occurring.
* **Uncertain environment** => partially observable environment or stochastic environment
* **Episodic environment** => when an agent performs an action based only on the current percept. Many classification tasks are episodic. For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions.
* **Sequential environment** => when an agent’s choice of action depends on its percept sequence and all actions performed so far.
* **Static environment** => while the agent thinks what action to take the environment state does not change. In other words, the agent must not worry about the passage of time. Crossword puzzles are static
* **Dynamic environment** => the agent must worry about the passage of time because the environment state changes while it thinks what action to take. Taxi driving is clearly dynamic
* **Semi-dynamic environment** =>when the environment does not change state while the agent thinks but its performance measures does. Chess, when played with a clock, is semi-dynamic
* **Discrete vs. continuous:** the discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent. For example, the chess environment has a finite number of distinct states (excluding the clock). Chess also has a discrete set of percepts and actions. Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.
* **Known environment** => when the agent knows what the possible outcomes of its actions are
* **Unknown environment** => when the agent does not known what the possible outcomes of its actions are

Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments. It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over. Conversely, an unknown environment can be fully observable—in a new video game, the screen may show the entire game state but I still don’t know what the buttons do until I try them.

## Structure of Agents

An Agent consists of two parts:

1. Architecture
2. Program

The Agent program is the software part of the agent which implements the AGENT FUNCTION.

The Agent Architecture, instead, is the physical device which runs the Agent Program and includes the sensors and the actuators.

## Different Kinds of Agents

1. Table-driven agent
2. Simple-reflex agent
3. Model-based reflex agent
4. Goal-based agent
5. Utility-based agent

## Table-driven agent

The table-driven agent makes use of a lookup table which maps a percept sequence to an action. Naturally, such an implementation is infeasible for real-world problems. The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behaviour from a smallish program rather than from a vast table.

A screenshot of a cell phone

Description automatically generated

## Simple reflex agent

These agents select actions on the basis of the current percept, ignoring the rest of the percept history.

Simple-reflex agents can be easily described through condition-action rules. In other words, given a certain state (condition), a certain action must be performed.

They are called simple reflex agents because they behave like reflexes in humans which given a certain condition spark off an action in our body.

Simple reflex agents are arguably very simple but this is why they are not rather intelligent. Indeed, if an environment is partially observable (it is not possible to extract all relevant information from the current state) then an agent should base its action on its percept sequence and not only on the current state.

In other words, they are suitable for fully observable environments but not for partially observable environments.

Suppose that a simple reflex vacuum agent is deprived of its location sensor and has only a dirt sensor. Such an agent has just two possible percepts: [Dirty] and [Clean]. It can Suck in response to [Dirty]; what should it do in response to [Clean]? Moving Left fails (forever) if it happens to start in square A, and moving Right fails (forever) if it happens to start in square B. Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments. Escape from infinite loops is possible if the agent can randomize its actions. For example, if the vacuum agent perceives [Clean], it might flip a coin to choose between Left and Right. It is easy to show that the agent will reach the other square in an average of two steps.

A screenshot of a cell phone

Description automatically generated

## Model-based reflex agent

Model-based reflex agents behave in the same way as simple reflex agents but unlike simple-reflex agents they are suitable for partially observable environments.

They are suitable for partially observable environments because they are based on a MODEL of the surrounding environment.

This MODEL describes to the agent how its surrounding environment evolves independently from the agent and how the agent’s actions affect the environment.

Thanks to the MODEL, even though the environment is partially observable, the agent can guess what the environment looks like by combining its percept sequence with the current state of the environment.

Therefore, a model based agent must keep an internal state which is not the current state of the environment but the best guess of the current environment state by combining the current state with the percept sequence and merge this together by using the laws described in the MODEL.

Regardless of the kind of representation used, it is seldom possible for the agent to determine the current state of a partially observable environment exactly.

Once the internal node is updated the agent behaves like a simple reflex agent by using its condition-actions rules.

A screenshot of a cell phone

Description automatically generated

## Goal-based agent

A goal-based agent is a model-based agent which performs its actions based on the current best guess of the state of the world and a goal.

Unlike, simple-reflex agents which base their choice of action on basic condition-action rules, goal-based agents base their choice of action on a given goal.

Notice that decision making of this kind is fundamentally different from the condition– action rules described earlier, in that it involves consideration of the future—both “What will happen if I do such-and-such?” and “Will that make me happy?”.

## Utility-based agents

Goals alone are not enough to generate high-quality behaviour in most environments. For example, many action sequences 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 binary distinction between “happy” and “unhappy” states. A more general performance measure should allow 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. We have already seen that a performance measure assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable ways of getting to the taxi’s destination. An agent’s utility function is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

Therefore, a utility-based agent, given the current state of the world (which is interpreted by the MODEL, so a utility-based agent is also a model-based agent ), performs the action which is expected to maximise the EXPECTED UTILITY.

A screenshot of a cell phone

Description automatically generated

## How to represent the environment

As we mentioned earlier, the axis along which atomic, factored, and structured repre sentations lie is the axis of increasing expressiveness. Roughly speaking, a more expressive representation can capture, at least as concisely, everything a less expressive one can capture, plus some more. Often, the more expressive language is much more concise; for example, the rules of chess can be written in a page or two of a structured-representation language such as first-order logic but require thousands of pages when written in a factored-representation language such as propositional logicA screenshot of a cell phone

Description automatically generated.

## Learning Agent

A learning agent is agent that is capable of augmenting or even replacing its prior knowledge by observing the outcome of its actions in relation to the environment.

A learning agent can assess the outcome of its action based on its PERFORMANCE MEASURE.

A screenshot of a cell phone

Description automatically generated

A learning agent can be divided up into 2 parts:

1. Performance element
2. Learning element
3. Problem generator
4. Critic element

The performance element is responsible for selecting external actions and 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. In other words, the learning elements is responsible for making improvements.

The last component of the learning agent is the problem generator. It is responsible for suggesting actions that will lead to new and informative experiences. The point is that if the performance element had its way, it would keep doing the actions that are best, given what it knows. But if the agent is willing to explore a little and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run. The problem generator’s job is to suggest these exploratory actions. This is what scientists do when they carry out experiments. Galileo did not think that dropping rocks from the top of a tower in Pisa was valuable in itself. He was not trying to break the rocks or to modify the brains of unfortunate passers-by. His aim was to modify his own brain by identifying a better theory of the motion of objects.

To make the overall design more concrete, let us return to the automated taxi example. The performance element consists of whatever collection of knowledge and procedures the taxi has for selecting its driving actions. The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes a quick left turn across three lanes of traffic, the critic observes the shocking language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installation of the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions.

The critic element evaluates the consequences of the actions of the learning agent using the PERFORMANCE MEASURE and passes in this information to the Learning element which acts consequently changing the performance element.