2.1 Agent, rational agent and environment

Agent

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators. An agent is embodied (i.e. situated) in an environment. An autonomous agent can be defined as "a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future". For example, a game-based agent is situated in a virtual game environment, whereas robotic agents are situated in a real (or possibly simulated) environment.

The term 'bot' – an abbreviation for robot – has become common as a substitute for the term 'agent'.

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.

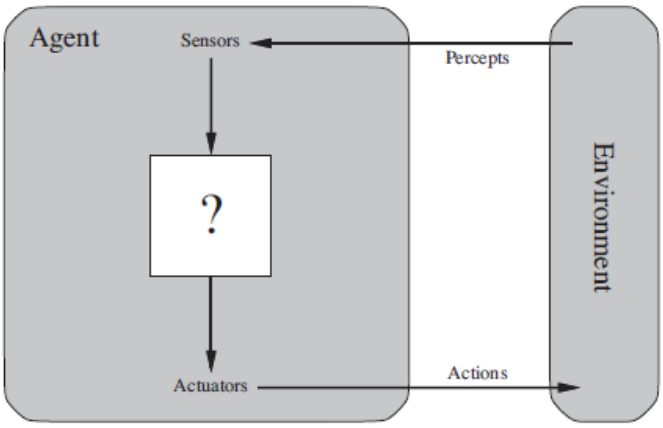

Figure 1. An agent

An agent's percept sequence is the complete history of everything the agent has ever perceived. In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived. Mathematically speaking, we say that an agent's behavior is described by the agent function that maps any given percept sequence to an action.

Rational Agent

A rational agent is one that does the right thing. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing? When an agent is placed 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. If we define success in terms of agent's opinion of its own performance, an agent could achieve perfect rationality simply by deceiving itself that its performance was perfect.

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.

Rational agent: *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.*

Intelligent agent

Intelligent agents sense their environments. The sensory data or percepts include not only data about other objects, but also about the impact of the agent itself on the state of affairs in the environment. The sensors could be organic, like eyes and ears and their neural processors, or artificial, like video and audio processors integrated into a digital computer. The environment could be a very limited domain, like a blocks world, or very sophisticated one, like the stock market or a set of asteroids. The sensors must be appropriate to the sort of objects with which the agent is designed to interact. Whatever the sensors, the history of all percepts sensed by the agent is critical to its future interaction with the environment.

The set of all percepts that an agent has to start with plus those gained by interaction with the environment (experience) is the percept sequence of the agent. An intelligent agent consults its percept sequence and the current state of affairs (which may be considered part of the total percept sequence) in light of its goals before taking action. This means that intelligent computer agents, like human agents, consult past experience and the current situation before deciding what course of action will further its goals.

In summary, intelligent agents are the autonomous agents that are capable of performing following tasks:

- Perceive the environment and respond to the change that occur in order to satisfy the design objective.
- Interact with other agents (and possibly humans) to satisfy design objective.

2.2 Environment and its properties

Environment

An environment is everything in the world that surrounds the agent that is not part of the agent itself. This is where the agent 'lives' or operates, and provides the agent with something to sense and somewhere for it to move around. An environment is similar to the world in real life having some of its properties.

To build a rational agents, we need to define "problems" to which rational agents are the "solutions". This is what a task environment is called. To define task environment of an agent, we have to specify the performance measure, the environment, and the agent's actuators and sensors. This specification is called the **PEAS (Performance, Environment, Actuators and Sensors)** description. In designing an agent, the first step must always be to specify the task environment as fully as possible.

For example, an automated taxi driver.

First, what is the **performance measure** to which we would like our automated driver to aim? 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. Obviously, some of these goals conflict, so tradeoffs will be required.

Next, what is the driving **environment** that the taxi will face? 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, and potholes. The taxi must also interact with potential and actual passengers. There are also some optional choices. 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 output to 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 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.

PEAS description of automated taxi driver and several other agents are as follows:

Agent Type	**Performance Measure**	**Environment**	**Actuator**	**Sensor**
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays

Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

Properties of task environment

An environment can have various attributes from the point of view of the agent. We can, however, identify a fairly small number of dimensions along which task environments can be categorized. These dimensions determine, to a large extent, the appropriate agent design and the applicability of each of the principal families of techniques for agent implementation.

- **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. In such environment, the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking. If the agent has no sensors at all then the environment is unobservable.
- **Single agent vs. multiagent:** If an environment is being acted upon by only one agent, such environment is single agent. Otherwise it is multiagent. The distinction between single-agent and multiagent environments may seem simple enough. For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment.

There are, however, some subtle issues. How does one agent (A) treat another agent (B)? The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior. 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. It is also partially competitive because, for example, only one car can occupy a parking space.

- **Deterministic vs. stochastic.** 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. In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment. If the environment is partially observable, however, then it could appear to be stochastic. Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic. Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning. The simple vacuum world is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism
- **Episodic vs. sequential:** In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. 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 and the current decision doesn't affect whether the next part is defective. In sequential environments, on the other hand, the current decision could affect all future decisions.3 Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences. Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.
- **Static vs. dynamic:** If the environment can change while an agent is acting, then we say the environment is dynamic for that agent; otherwise, it is static. Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action,

nor need it worry about the passage of time. Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semidynamic. Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static.

- **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. 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.).

So, we can say that the hardest case is partially observable, multiagent, stochastic, sequential, dynamic and continuous. Taxi driving is hard in all these senses.

2.3 The structure of agents

The job of AI is to design an agent program that implements the agent function the mapping from percepts to actions. It is assumed that this program will run on some sort of computing device with physical sensors and actuators.

agent = architecture + program

Each kind of agent program combines particular components in particular ways to generate actions. There are five basic kinds of agent programs that embody the principles underlying almost all intelligent systems:

- Simple reflex agents
- Model-based reflex agents
- Goal-based agents
- Utility-based agents
- Learning agents

2.3.1 Simple reflex agent

Simple reflex agents ignore the rest of the percept history and act only on the basis of the current percept. The agent function is based on the condition-action rule. A condition-action rule is a rule that maps a state i.e, condition to an action. If the condition is true, then the action is taken, else not. This agent function only succeeds when the environment is fully observable.

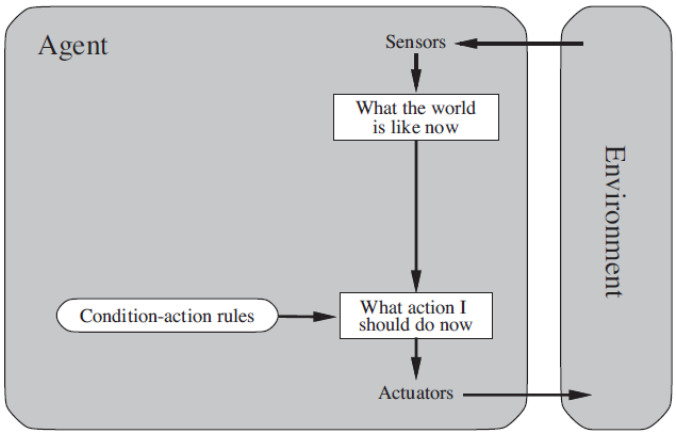

Figure 2. Simple Reflex Agent

For example in the taxi triving agent's case, if the car in front brakes and its brake lights come on, then braking should be initiated.

This connection is called a condition–action rule, written as:

if car-in-front-is-braking ***then*** initiate-braking.

Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence.

2.3.2 Model based reflex agents

The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now. That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state. Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.

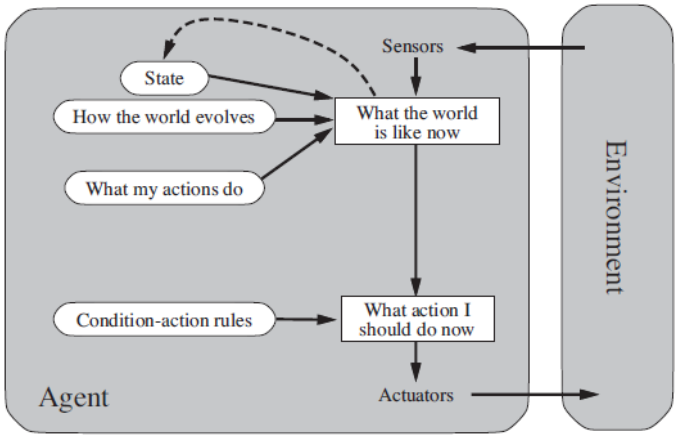

Figure 3. A model-based reflex agent

- information about how the world evolves independently of the agent
- information about how the agent's own actions affect the world

This knowledge about "how the world works is called a model of the world. A model based agent uses such model.

2.3.3 Goal Based Agent

Knowing something 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, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to. In other words, as well as a current state description, the GOAL agent needs some sort of goal information that describes situations that are desirable: for example, being at the passenger's destination. The agent program can combine this with the model (the same information as was used in the modelbased reflex agent) to choose actions that achieve the goal.

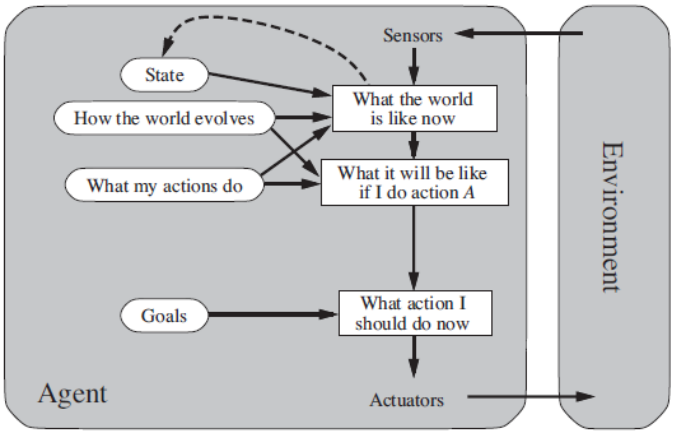

Figure 4. A model-based, goal-based agent

The knowledge that supports its decisions is represented explicitly and can be modified, which makes these agents more flexible. They usually require search and planning. The goal based agent's behavior can easily be changed. For example if it starts to rain, the agent can update its knowledge of how effectively its brakes will operate; this will automatically cause all of the relevant behaviors to be altered to suit the new conditions.

2.3.4 Utility Based Agents

Goals alone are not enough to generate high-quality behavior 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. Term utility refers to this degree of happiness.

The utility-based agent will rate each scenario to see how well it achieves certain criteria with regard to the production of a good outcome. Things like the probability of success, the resources needed to execute the scenario, the importance of the goal to be achieved, the time it will take, might all be factored in to the utility function calculations.

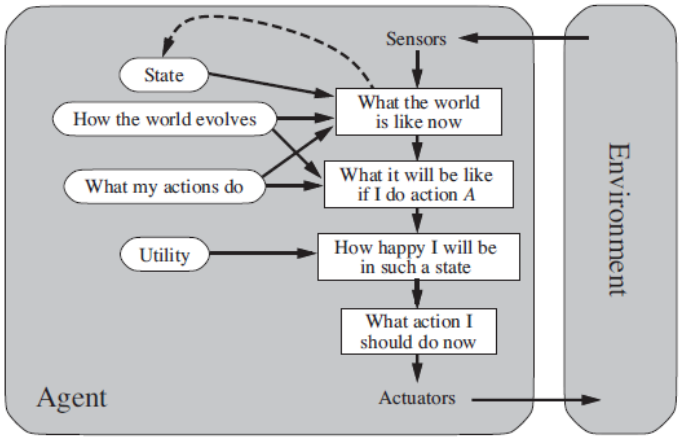

Figure 5. A model-based, utility-based agent.

2.3.5 Learning Agents

In his famous early paper, Turing (1950) considers the idea of actually programming his intelligent machines by hand. He estimates how much work this might take and concludes "Some more expeditious method seems desirable." The method he proposes is to build learning machines and then to teach them. In many areas of AI, this 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 competent than its initial knowledge alone might allow.

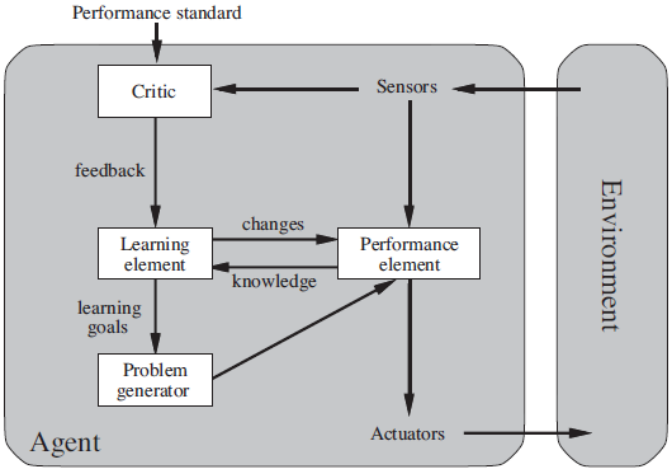

Figure 6. A general learning agent.

A learning agent can be divided into four conceptual components. The most important distinction is between the learning element, which is responsible for making improvements, and the performance element, which is responsible for selecting external actions. The performance element 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. The problem generator 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

Learning in intelligent agents can be summarized as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent.

References

[1] W. J. Teahan, Artificial Intelligence – Agents and Environments, Frederiksberg: Ventus Publishing ApS, 2010.

[2] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, New Jersey: Pearson Education, Inc., 2010.