Agents and environment

Artificial intelligence

Francesco Corona

Intelligent agents

We have identified the concept of **rational agents** as central to our approach to artificial intelligence

- Now, we make this notion more concrete

We shall see that the concept of rationality can be applied to a wide variety of agents operating in any imaginable environment

We use this concept to develop a set of principles for designing successful agents, systems that can reasonably be called intelligent

- We begin by examining **agents** and **environments**
- Then, we discuss the **coupling** between them

Intelligent agents (cont.)

The observation that some agents behave better than others leads to the idea of a rational agent, one that behaves as well as possible

How well an agent can behave depends on the **environment**

- As, some environments are more difficult than others

We give a crude categorisation of environments and show how their properties influence the design of suitable agents for them

We shall also describe a number of basic 'skeleton' agent designs

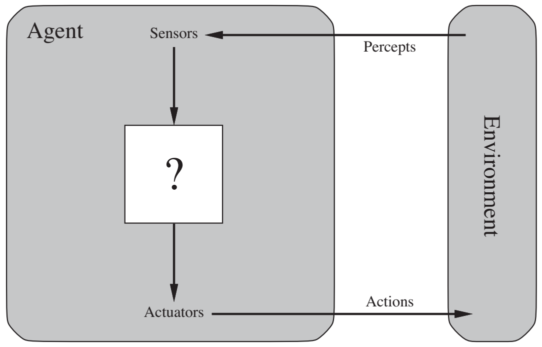
Agents and environment

Intelligent agents

Agents and environment

Definition

An **agent** is anything viewed as **perceiving** its environment thru **sensors** and **acting** upon that environment through **actuators**



Agents and environment (cont.)

Example

- 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, network packets as sensory inputs and acts on the environment by displaying on screen, writing files, sending network packets

Agents and environment (cont.)

We use the term **percept** to refer to the agent's perceptual inputs

- at any given instant

A **percept sequence** is the history of everything ever perceived

Remark

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 has NOT perceived

By specifying the agent's choice of action for all possible percept sequences, we said $\pm$ everything there is to say about the agent

Definition

- Mathematically, an agent's behaviour is described by the **agent function** that maps any percept sequence to an action

Agents and environment (cont.)

Imagine tabulating the agent function that describes any agent

- For most agents, this would be a very large table

The table is in fact infinite, unless we place a bound on the length of percept sequences we want to consider

In principle, given an agent to experiment with, we can construct this table by trying out all possible percept sequences and recording which actions the agent does in response

- The table is an external characterisation of the agent

Definition

Internally, the agent function for an artificial agent will be implemented by an **agent program**

Agents and environment (cont.)

It is important to keep these two ideas distinct:

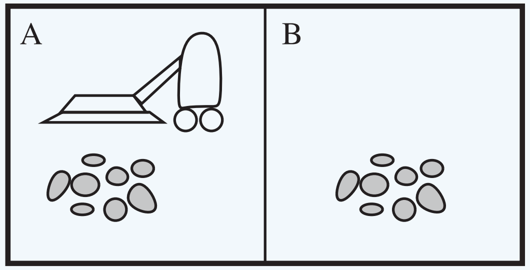
- The agent function is an abstract mathematical description;
- The agent program is a concrete implementation, running within some physical system

Agents and environment (cont.)

Example

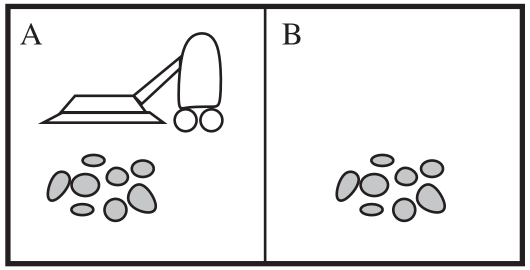
The vacuum-cleaner and its world

It is so simple that we can describe everything that happens



It is a made-up world, so we can invent many variations

This particular world has two locations: squares A and B



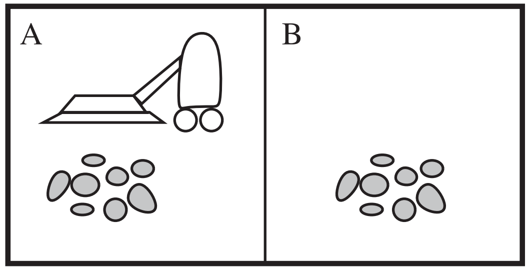
The vacuum agent perceives which square it is in and whether there is dirt in the square

- It can choose to move left or right, suck up dirt or do nothing

One very simple agent function is

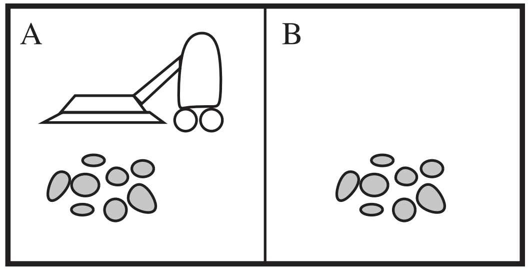
- if current square (location) is dirty (status), then suck (action);
- otherwise, move to the other square (action)

Agents and environment (cont.)



Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

Agents and environment (cont.)



```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Agents and environment (cont.)

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$

Various vacuum-cleaner agents can be defined by filling the 'action' column in various ways

- What is the right way to fill out the table?

We are asking what makes an agent good/bad, intelligent/dumb?

Agents and environment (cont.)

Remark

The notion of agent is meant to be a tool for analysing systems

- It is not an absolute characterisation that divides the world into agents and non-agents

One can view a hand-held calculator as an agent that chooses the action of displaying '4' when given the percept sequence '2+2='

- Such an analysis does not aid our understanding of the calculator

Agents and environment (cont.)

In a sense, all areas of engineering can be seen as designing artefacts that interact with the world

- AI operates at the most interesting end of the spectrum, where the artefacts have significant computational resources and the task environment requires nontrivial decision making

Rationality and good behaviour

Intelligent agents

Rationality and good behaviour

A **rational agent** is defined as one that **does the right thing**

- conceptually speaking, every entry in the table for the agent function is filled out correctly

Doing the right thing is better than doing the wrong thing (uh?)

- But, what does it mean to do the right thing?

This age-old question can be answered in an age-old way

- By considering the consequences of the agent's behaviour

Rationality and good behaviour (cont.)

An agent in an environment generates a sequence of actions

- according to the percepts it receives

Sequence of actions $\Rightarrow$ Environment thru a sequence of states[1]

- If the sequence is desirable, the agent performed well

This notion of desirability is captured by a **performance measure**

- It evaluates any given sequence of environment states

[1]Note that we said environment states, not agent states. If we define success in terms of agent's opinion of its own performance, an agent could achieve perfect rationality simply by deluding itself that its performance was perfect. Human agents in particular are notorious for 'sour grapes', believing they did not really want something (e.g., a Nobel Prize) after not getting it

Rationality and good behaviour (cont.)

Various fixed performance measure for all tasks and agents

- The designer devise one appropriate to the circumstances
- This is not as easy as it sounds

Remark

'*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*'

Rationality and good behaviour (cont.)

Example

Consider the vacuum-cleaner agent and measure performance by the amount of dirt cleaned up in a single eight-hour shift

- With a rational agent, what you ask for is what you get

It can maximise this performance measure by cleaning up the dirt, 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

Award one point for each clean square, at each time step

- with penalty for electricity consumed and noise generated

Rationality and good behaviour (cont.)

Even when obvious pitfalls are avoided, there remain some issues

- For instance, the notion of 'clean floor' is based on average cleanliness over time

The same average cleanliness can be achieved by different agents

- One that does a mediocre job all the time while the other cleans energetically but takes long breaks

Which is preferable seems to be a fine point of janitorial science, but it is a deep philosophical question with profound 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 mild poverty, or one in which some live in plenty while others are very poor?

Rationality

Rationality and good behaviour

Rationality

What is rational at any given time depends on four things

1. The performance measure that defines the criterion of success
2. The agent's prior knowledge of the environment
3. The actions the agent can perform
4. The agent's percept sequence to date

Definition

This leads to the **definition of a rational agent**

- '*For each possible percept sequence, a rational agent should select an action that is expected to maximise its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has*'

Rationality (cont.)

Example

The vacuum-cleaner agent that cleans a square
if it is dirty and moves to the other square if not

- Is this a rational agent? That depends!

We need to say what the performance measure is, what is known about the environment, and what are the sensors and actuators

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

Rationality (cont.)

Let us assume the following:

Assumption

- The performance measure awards ‘one point, for each clean square, at each time step, over a ‘lifetime’ of 1K time steps’
- The ‘geography’ of the environment is known *a priori* but the dirt distribution and the initial location of the agent are not
- Clean squares stay clean and `Sucking` cleans current square
- The `Left` and `Right` actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is
- The only available actions are `Left`, `Right`, and `Suck`
- The agent correctly perceives its location and whether that location contains dirt

Rationality (cont.)

Under these circumstances, we claim that **the agent is rational**

- Its expected performance is at least as high as any other agent’s

Exercise

Show that the vacuum-cleaner agent function

Percept sequence	Action
$[A, Clean]$	$Right$
$[A, Dirty]$	$Suck$
$[B, Clean]$	$Left$
$[B, Dirty]$	$Suck$
$[A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Dirty]$	$Suck$
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	$Right$
$[A, Clean], [A, Clean], [A, Dirty]$	$Suck$
⋮	⋮

is indeed rational under the assumptions

Rationality (cont.)

It suffices to show that for all possible actual environments (all dirt distributions and initial locations), this agent cleans the squares at least as fast as any other agent

- This is trivially true when there is no dirt
- When there is dirt in the initial location and none in the other location, the world is clean after one step (no agent can do better)
- When there is no dirt in the initial location but dirt in the other, the world is clean after two steps (no agent can do better)
- When there is dirt in both locations, the world is clean after three steps (no agent can do better)

(Note: in general, the condition stated in the first sentence of this answer is much stricter than necessary for an agent to be rational)

Rationality (cont.)

The same agent would be irrational under different circumstances

For example,

- once all the dirt is cleaned up, the agent will oscillate needlessly back and forth;
- if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly

A better agent for this case would do nothing, once it is sure that all the squares are clean

Rationality (cont.)

If clean squares can become dirty again, the agent should check and re-clean them if needed, occasionally

If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B

Omniscience, learning and autonomy

Rationality and good behaviour

Omniscience, learning and autonomy

We carefully distinguish between rationality and **omniscience**

- An omniscient agent knows the actual outcome of its actions and can act accordingly

Omniscience is impossible in reality

Example

I am walking along the Champs Elysès one day and I see an old friend across the street

There is no traffic nearby and I am not otherwise engaged

- So, being rational, I start to cross the street

Meanwhile, at 33K feet, a cargo door falls off a passing airliner, and before I make it to the other side of the street I am flattened

Was I irrational to cross the street?

Omniscience, learning and autonomy (cont.)

The example shows that rationality is not the same as perfection

Remark

- **Rationality maximizes expected performance**
- **Perfection maximizes actual performance**

Retreating from a requirement of perfection
is not just a question of being fair to agents

The point is that if we expect an agent to do what turns out to be best action *after the fact*, it will be impossible to design agents

- Impossible to fulfil this specification (unless we improve the performance of crystal balls or time machines)

Omniscience, learning and autonomy (cont.)

Our definition of rationality does not require omniscience, because the rational choice depends only on the percept sequence to date

- We must also ensure that we have not inadvertently allowed the agent to engage in decidedly under-intelligent activities

Omniscience, learning and autonomy (cont.)

Example

If an agent does not look both ways before crossing, its percept sequence will not tell it that there is a truck approaching fast

Does our definition of rationality say that it is now OK to cross?

- First, it would not be rational to cross the road given this uninformative percept sequence: the risk of accident from crossing without looking is too great
- Second, a rational agent should choose the 'looking' action before stepping into the street, because looking helps maximize the expected performance

Omniscience, learning and autonomy (cont.)

Doing actions in order to modify future percepts, something that is called **information gathering**, is an important part of rationality

Example

- A second example of information gathering is provided by the **exploration** that must be undertaken by a vacuum-cleaning agent in an initially unknown environment

Omniscience, learning and autonomy (cont.)

Our definition requires a rational agent not only to gather info but also to learn as much as possible from what it perceives

Remark

The agent's initial config could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented

Omniscience, learning and autonomy (cont.)

Extreme case: The environment is completely known a priori

- The agent need not perceive or learn
- It simply acts correctly

Such agents are fragile

Example

The dung beetle

After digging its nest and laying its eggs, it fetches a ball of dung from a nearby heap to plug the entrance

- If the ball of dung is removed from its grasp *en route*, the beetle continues its task and pantomimes plugging the nest with the nonexistent dung ball, never notices it is missing

Evolution has built an assumption into the beetle's behaviour, and when it is violated, unsuccessful behaviour results

Example

The sphex wasp: Slightly more intelligent

- 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 egg
- The caterpillar serves as a food source when the eggs hatch

If the caterpillar is moved a few inches away while the sphex is doing the check, it will revert to the 'drag' step of its plan

- It 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

- Thus, will not change it

Omniscience, learning and autonomy (cont.)

Definition

Autonomy (or, lack thereof): The extent that an agent relies on the prior knowledge of its designer rather than on its own percepts

- A rational agent should be autonomous, it should learn what it can to compensate for partial or incorrect prior knowledge

Omniscience, learning and autonomy (cont.)

Example

A vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not

Practically, one seldom requires complete autonomy from start

- When the agent has had little or no experience, it would have to act randomly unless the designer gave some assistance

Omniscience, learning and autonomy (cont.)

It would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn

- After sufficient experience of its environment, the behaviour of a rational agent can become effectively *independent of its prior knowledge*

Hence, the incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments

The environment
Intelligent agents

The environment

Now that we have a definition of rationality, we are almost ready to think about building rational agents

- First, however, we must think about **task environments**
- ‘Problems’ to which rational agents are ‘solutions’

We begin by showing how to **specify a task environment**, illustrating the process with a number of examples

We show that task environments come in a **variety of flavours**

- The flavour of the task environment directly affects the appropriate design for the agent program

Specification of task environments
The environment

Specification of task environments

Definition

When discussing rationality (vacuum-cleaner), we had to specify performance measure, environment, and actuators and sensors

- We group them under the heading of the **task environment**

PEAS: **Performance**, **Environment**, **Actuators**, **Sensors**

Remark

In designing an agent, the first step must always be to specify the task environment as fully as possible

The vacuum world was a simple example

Specification of task environments

Example

An automated taxi driver: This is a more complex problem

- ‘*We should point out, before the reader becomes alarmed, that a fully automated taxi is currently somewhat beyond the capabilities of existing technology*’ :P

The full driving task is open-ended, as there is no limit to the novel combinations of circumstances that can arise

Agent Type	Performance Measure	Environment	Actuators	Sensors
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

The PEAS description for the taxi’s task environment

Specification of task environments (cont.)

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, GPS, odometer, accelerometer, engine sensors, keyboard

Performance measure the automated driver should aspire to?

- Getting to the correct destination;
- Minimising fuel consumption and wear and tear;
- Minimising the trip time or cost;
- Minimising violations of traffic laws
  and disturbances to other drivers;
- Maximising safety and passenger comfort;
- Maximising profits

Remark

Some of these goals conflict, so tradeoffs will be required

Specification of task environments (cont.)

Agent Type	Performance Measure	Environment	Actuators	Sensors
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

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 interact with potential/actual passengers

Specification of task environments (cont.)

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 GB or JAP

Remark

The more restricted the environment, the easier the design problem

Specification of task environments (cont.)

Agent Type	Performance Measure	Environment	Actuators	Sensors
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

Actuators for an automated taxi are those available to humans

- Control over the engine through the accelerator
  and control over steering and braking

In addition, it will need output to a display or voice synthesiser

- to talk back to the passengers, and perhaps some way to communicate with other vehicles, politely or otherwise

Specification of task environments (cont.)

Agent Type	Performance Measure	Environment	Actuators	Sensors
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

The basic **sensors** for the taxi will include:

- One or more controllable videocameras, to see the road;
- Infrared or sonar sensors, to detect distances to other cars and obstacles
- To avoid speeding tickets, the taxi should have a speedometer
- To control the vehicle properly, especially on curves, it should have an accelerometer

Specification of task environments (cont.)

To determine the mechanical state of the vehicle, it will need the usual array of engine, fuel, and electrical system sensors

Like humans, 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.

Specification of task environments (cont.)

The set of basic PEAS elements vary with agent typology and task

Agent Type	Performance Measure	Environment	Actuators	Sensors
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 environments

The environment

Properties of task environments

The range of task environments that might arise in AI is vast

- We can, however, identify a small number of dimensions along which task environments can be categorised

These dimensions determine, largely, the right agent design and applicability of the main families of implementation techniques

- The definitions here are informal

Properties of task environments (cont.)

Definition

Fully observable v partially observable: If an agent's sensors give it access to the complete state of the environment at each point in time, then the task environment is **fully observable**

A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action

- Relevance, in turn, depends on the performance measure

Remark

Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world

Properties of task environments (cont.)

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

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, then the environment is **unobservable**

- One might think that in such cases the agent's plight is hopeless, but the agent's goals may still be achievable, sometimes with certainty

Properties of task environments (cont.)

Definition

Single agent v multi-agent: The distinction between single- and multi-agent environments may seem simple enough

Example

- An agent solving a cross-word puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment

There are some subtle issues to be considered

Properties of task environments (cont.)

First, we have described how an entity may be viewed as an agent, but we have not explained which entities must be viewed as agents

- Does an agent A (say, the taxi driver) have to treat an object B (another vehicle) as an agent, or can it be treated merely as an object behaving according to the laws of physics?

Remark

The key distinction is whether B's behaviour is best described as maximising a performance measure, whose value depends on agent A's behaviour

Properties of task environments (cont.)

Example

In chess, 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 **multi-agent environment**

Example

In the taxi-driving environment, avoiding collisions maximizes the performance measure of all agents

- It is a partially cooperative **multi-agent environment**
- It is also a partially competitive (only one car can occupy a parking space) **multi-agent environment**

Properties of task environments (cont.)

The agent-design problems in multi-agent environments are quite different from those in single-agent environments

- Communication often emerges as a rational behaviour in multi-agent environments;
- In some competitive environments, randomised behaviour is rational because it avoids the pitfalls of predictability

Properties of task environments (cont.)

Definition

Deterministic v stochastic: If the next state of the environment is completely determined by the current state and 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[2]

- If the environment is partially observable, however, then it could appear to be stochastic

[2] In our definitions, we ignore uncertainty that arises purely from the actions of other agents in a multi-agent environment (a game can be deterministic even though each agent may be unable to predict the actions of the others)

Properties of task environments (cont.)

Most real situations are so complex that it is not possible to keep track of all the unobserved aspects

- For practical purposes, they must be treated as stochastic

Example

- Taxi driving is clearly stochastic in this sense, because one can never predict the behaviour of traffic exactly (moreover, one's tires blow out and one's engine seizes up without warning)
- The vacuum world as we described it is deterministic (but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism)

Properties of task environments (cont.)

Definition

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

Properties of task environments (cont.)

The word 'stochastic' generally implies that uncertainty about outcomes is quantified in terms of probabilities

- a non-deterministic environment is one in which actions are characterised by their *possible* outcomes, but no probabilities are attached to them

Remark

Non-deterministic environment descriptions are usually associated with performance measures that require the agent to succeed, for *all possible* outcomes of its actions

Properties of task environments (cont.)

Definition

Episodic v 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

In a **sequential** environment, on the other hand, the current decision could affect all future decisions

Properties of task environments (cont.)

Example

Many classification tasks are episodic

- An agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions;
- Moreover, the current decision doesn't affect whether the next part is defective

- Chess and taxi driving are sequential: In both cases, short-term actions can have long-term consequences

Properties of task environments (cont.)

Remark

Episodic environments are simpler than sequential environments, because the agent does not need to think ahead

Properties of task environments (cont.)

Definition

Static v dynamic: If the environment can change while an agent is deliberating, then 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 has not decided yet, that counts as deciding to do nothing

Properties of task environments (cont.)

If the environment itself does not change with passage of time but agent's performance score does, the environment is **semi-dynamic**

Example

- 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 semi-dynamic
- Crossword puzzles are static

Properties of task environments (cont.)

Definition

Discrete v 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

Example

Chess 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

- Speed and location of the taxi and of the other vehicles take 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

Properties of task environments (cont.)

Definition

Known v unknown: Strictly speaking, this distinction refers not to the environment itself but to the agent's (or designer's) state of knowledge about the 'laws of physics' of the environment

- In a **known** environment, outcomes (or outcome probabilities if the environment is stochastic) for all actions are given

If the environment is **unknown**, the agent will have to learn how it works in order to make good decisions

Properties of task environments (cont.)

The distinction between known/unknown environments is not the same as the one between fully-/partially-observable environments

It is possible for a known environment to be partially observable

- 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 do not know what the buttons do until I try them)

Properties of task environments (cont.)

Remark

As expected, the hardest case is partially observable, multi-agent, stochastic, sequential, dynamic, continuous, and unknown

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

The agents
Intelligent agents

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

The agents

So far we have talked about agents by describing behaviour, that is the action that is performed after any given sequence of percepts

- Now we talk about how the insides work

The job of AI is to design an **agent program** that implements the agent function, the **mapping from percepts to actions**

Assumption

We assume this program will operate on some sort of computing device with physical sensors and actuators

- We call this the **architecture**
- **agent** $=$ **architecture** $+$ **program**

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

The agents (cont.)

The program is chosen appropriate for the architecture

- In general, the architecture makes the percepts from the sensors available to the program
- In general, the architecture feeds the program's actions to the actuators as they are generated

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Agent programs
The agents

Agent programs

The agent programs that we discuss all have the same skeleton:

- They take the current percept as input from the sensors and return an action to the actuators

Remark

Notice the difference between agent program and agent function

- the **agent program** takes the **current percept** as input
- the **agent function** takes the entire **percept history**

The agent program takes just the current percept as input because nothing more is available from the environment

- if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts

We describe the agent programs in simple pseudocode language

Agent programs (cont.)

Example

A trivial agent program that keeps track of the percept sequence and uses it to index into a table of actions to decide what to do

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

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

The **TABLE-DRIVEN-AGENT** program is invoked for each new percept and returns an action each time

- It retains the complete percept sequence in memory

Agent programs (cont.)

The table represents explicitly the agent function that the agent program embodies

- To build a rational agent in this way, we must construct a table that contains the appropriate action
- for every possible percept sequence

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$

Agent programs (cont.)

A table-driven approach to agent construction is doomed to fail

- Let $\mathcal{P}$ be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive)
- The lookup table will contain $\sum_{t=1}^{T} |\mathcal{P}|^t$ entries

Example

Consider an automated taxi with visual input from a single cam

- Stuff comes in at the rate of $\sim$ 27MBs[3]
- One hour drive: A lookup table with $+10^{250,000,000,000}$ entries

A lookup table for chess (a tiny, well-behaved fragment of world)

- At least 10^{150} entries

[3] 30fps, 640 $\times$ 480 pixels with 24-bit colour information

Agent programs (cont.)

The daunting size of these tables means that

- no physical agent in this universe will have the space to store the table
- the designer would not have time to create the table
- no agent could ever learn all the right table entries from its experience
- even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries

Example

Despite all this, TABLE-DRIVEN-AGENT does do what we want:

- it implements the desired agent function

Agent programs (cont.)

A key challenge: Find how to write programs that, to the extent possible, produce rational behaviour from a smallish program

- rather than from a vast table

Examples show that this can be done successfully in many areas

Example

- The huge tables of square roots used by engineers and school children prior to the 1970s have now been replaced by a five-line program for Newton's method running on PCs

Agent programs (cont.)

The four basic kinds of agent programs that embody the principles underlying almost all intelligent systems are

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

Each kind of agent program combines particular components in particular ways to generate actions

Agent programs (cont.)

How to convert these agents into learning agents that can improve performance of their components so as to generate better actions

- We shall describe a variety of ways in which components themselves can be represented within the agent

Simple reflex agents
The agents

Simple reflex agents

The simplest kind of agent is the **simple reflex agent**

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

Simple reflex agents (cont.)

Example

The vacuum agent is a simple reflex agent, its decision is based only on current location and on whether that location contains dirt

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
$\vdots$	$\vdots$

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
    if status = Dirty then return Suck
    else if location = A then return Right
    else if location = B then return Left
```

Simple reflex agents (cont.)

The agent program is small compared to the corresponding table

- The largest reduction comes from ignoring the percept history, which cuts down the number of possibilities from 4^T to 4
- A smaller reduction comes from the fact that when current square is dirty, the action does not depend on the location

Simple reflex agents (cont.)

Simple reflex behaviours occur even in more complex environments

Example

It is easy to imagine yourself as the driver of the automated taxi

- If the car in front brakes and its brake lights come on, then you should notice this and initiate braking

Processing is done on the visual input to establish the condition '`The car in front is braking`' and this triggers established connection in the agent program to action '`Initiate braking`'

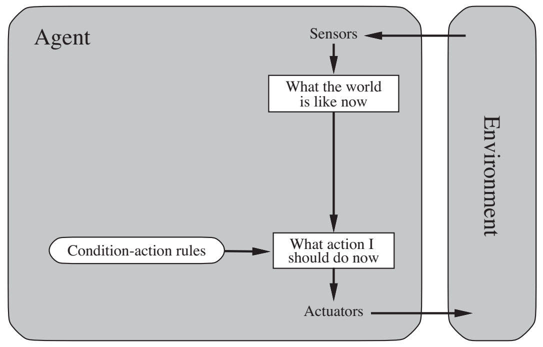
We call such a connection a **condition-action rule** written as

if `car-in-front-braking` **then** `initiate-braking`

Humans have many such connections some of which are learned (car braking) and some others are innate reflexes (blinking)

Simple reflex agents (cont.)

A general and flexible approach to design simple reflex agents is first to build a general-purpose interpreter for condition-action rules and then to create rule sets for specific task environments



Simple reflex agents (cont.)

The schematic of this general program shows how condition-action rules allow to make the connection from percept to action

- Rectangles denote the current internal state of the agent's decision process
- Ovals represent the background information used in the decision process

Simple reflex agents (cont.)

Example

The simple reflex agent acts according to the rule whose condition matches the current state, as defined by the percept

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

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

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

Simple reflex agents (cont.)

Simple reflex agents have the admirable property of being simple

- but they turn out to be of limited intelligence

Remark

The simple reflex agent works only if the correct decision can be made on the basis of only the current percept

- Only if the environment is fully observable

Even a tiny bit of un-observability can be cause of serious troubles

Simple reflex agents (cont.)

Example

- The braking rule assumes condition `car-in-front-braking` to be determined from current percept, a single frame of video

This works if the car in front has a centrally mounted brake light

- Unfortunately, older models have different configurations of taillights, brake lights, and turn-signal lights
- It is not always possible to tell from a single image whether the car is braking

A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all

Simple reflex agents (cont.)

Example

Issues arise with a vacuum agent if we suppose that a simple reflex agent has lost its location sensor and has only a dirt sensor

Such an agent has 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
- Moving `Right` fails (forever) if it happens to start in square B

Simple reflex agents (cont.)

Percept sequence	Action
[*A*, *Clean*]	*Right*
[*A*, *Dirty*]	*Suck*
[*B*, *Clean*]	*Left*
[*B*, *Dirty*]	*Suck*
[*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Clean*]	*Right*
[*A*, *Clean*], [*A*, *Clean*], [*A*, *Dirty*]	*Suck*
⋮	⋮

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

Simple reflex agents (cont.)

For simple reflex agents that operate in partially observable environments, infinite loops are often unavoidable

Escape from infinite loops is possible, if the agent can randomise its actions

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.

Then, if that square is dirty, the agent will clean it

- the task will be complete

Simple reflex agents (cont.)

Randomised simple reflex agents might be able to outperform deterministic simple reflex agents

Randomised behaviour of the right kind can be rational in some multi-agent environments

In single-agent environments, randomisation is usually not rational

Remark

It is a useful trick that helps a simple reflex agent in some cases

- In most situations we can do much better with more sophisticated deterministic agents

Model-based reflex agents
The agents

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 NOT see now

- 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

Example

For the braking problem, internal state is not too extensive

The previous frame from camera, allowing the agent to detect when two red lights at the vehicle edge go on or off simultaneously

For other driving tasks such as changing lanes, the agent needs to keep track of where other cars are, if it cannot see them all at once

Model-based reflex agents (cont.)

Updating the 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[4]
- Second, we need some information about how the agent's own actions affect the world[5]

Knowledge about 'how the world works', whether implemented as Boolean circuits or principled theories, is a **model** of the world

- An agent that uses such a model is a **model-based agent**

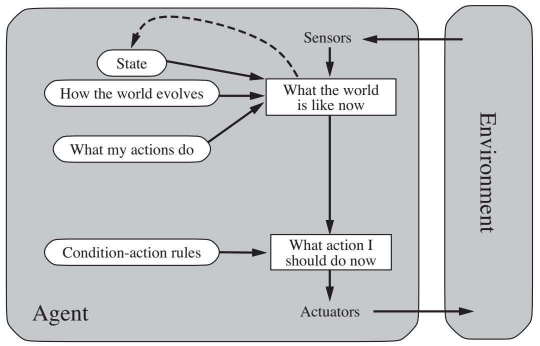
[4] For example, an overtaking car generally will be closer than it was before

[5] For example, when the agent steers clockwise the car turns to the right, or after driving for five minutes northbound on the freeway one is usually about five miles north of where one was five minutes ago

Model-based reflex agents (cont.)

The structure of the model-based reflex agent with internal state shows how the current percept is combined with the old internal state to generate the updated description of the current state

- based on the agent's model of how the world works



Model-based reflex agents (cont.)

A model-based reflex agent keeps track of the current state of the world using an internal model and then it chooses an action

- the choice is done in the same way as the reflex agent

Example

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of 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(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

Interesting part of agent program: Function **UPDATE-STATE**

- It is responsible for creating the new internal state description

Model-based reflex agents (cont.)

The details of how models and states are represented vary widely depending on type of environment and technology used in design

Remark

Whatever the kind of representation, it is seldom possible for the agent to determine THE state of a partially observable environment

- The box labeled 'what the world is like now' represents the agent's 'best guess(es)'

Example

An automated taxi may not be able to see around the large truck that has stopped in front of it and can only guess about what may be causing the hold-up. Thus, uncertainty about the current state may be unavoidable, but the agent still has to make a decision

Model-based reflex agents (cont.)

The internal 'state' maintained by a model-based agent does not have to describe 'what the world is like now' in a literal sense

Example

A taxi may be driving back home, and it may have a rule telling it to fillup with gas on the way home unless it has at least 50% tank

- Although 'driving back home' may seem to be an aspect of the world state, the fact of the taxi's destination is actually an aspect of the agent's internal state

If you find this puzzling, consider that the taxi could be in exactly the same place at the same time, but towards another destination

Goal-based agents
The agents

Goal-based agents

Knowing something about the current state of the environment is not always enough to decide what to do

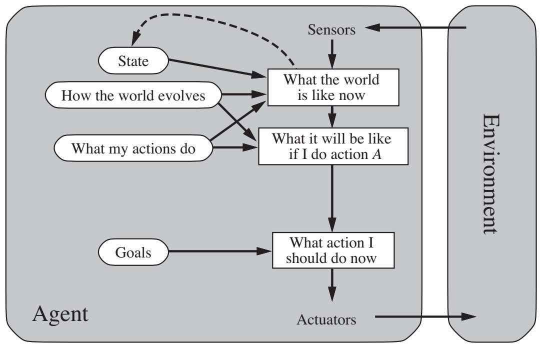
Example

- At a road junction, the taxi can turn left, right, or go straight
- The correct decision depends on where it is trying to get to

In other words, as well as a current state description, the agent needs some sort of **goal** information describing desirable situations

- The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal

Goal-based agents (cont.)


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

Goal-based agents (cont.)

Sometimes goal-based action selection is straightforward

- When goal satisfaction results straight from a single action

Sometimes goal-based action selection will be more tricky

- For example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal

Definition

Search and **planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals

Goal-based agents (cont.)

Decision making of this kind is deeply different from condition-action rules

- It involves consideration of the future ('What will happen if I do such-and-such?' and 'Will that make me happy?')

In reflex agent designs this explicit information is not represented, because its built-in rules map directly from percepts to actions

- The reflex agent brakes when it sees brake lights

A goal-based agent, in principle, could reason that if the car in front has its brake lights on, it will slow down

- Given the way the world usually evolves, the only action that will achieve the goal of not hitting other cars is to brake

Goal-based agents (cont.)

Although the goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified

- If it starts to rain, then the agent can update its knowledge of how effectively its brakes will operate

This will automatically cause all relevant behaviours to be altered to suit the new conditions

- For the reflex agent, on the other hand, we would have to rewrite many condition-action rules

Goal-based agents (cont.)

The goal-based agent's behaviour can easily be changed to go to a different destination, by specifying that destination as the goal

Example

- The reflex agent's rules for when to turn and when to go straight will work only for a single destination
- They must all be replaced to go somewhere new

Utility-based agents
The agents

Utility-based agents

Goals alone are not enough to generate high-quality behaviour

Example

Many action sequences will get the taxi to destination (the goal) but some are quicker, safer, more reliable, or cheaper than others

Goals provide a binary distinction between 'happy/unhappy' states

A more general performance measure should allow a comparison of world states according to how happy they would make the agent

- Because 'happy' does not sound very scientific, economists and computer scientists use the term **utility** instead

Utility-based agents (cont.)

A performance measure assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable ones (say, ways of getting to the taxi's destination)

Definition

A **utility function** is kind of an internalisation of the performance

- If internal utility function and external performance measure are in agreement, then an agent that chooses actions to maximise its utility will be rational according to performance

Utility-based agents (cont.)

Let us emphasise again that this is not the only way to be rational

- we have already seen a rational agent program for the vacuum world that has no idea what its utility function is

Like goal-based agents, a utility-based agent has many advantages

- Flexibility and learning

Furthermore, in two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions

- First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff
- 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 against the importance of the goals

Utility-based agents (cont.)

Partial observability and stochasticity are ubiquitous in the real world, and so, therefore, is decision making under uncertainty

- Technically, a rational utility-based agent chooses the action that maximises the **expected utility** of the action outcomes
- Expected utility is the utility the agent expects to derive, on average, given the probabilities and utilities of each outcome

It is possible to show that any rational agent must behave as if it has a utility function whose expected value it tries to maximise

- An agent that possesses an explicit utility function can make rational decisions with a general-purpose algorithm that does not depend on the specific utility function being maximised

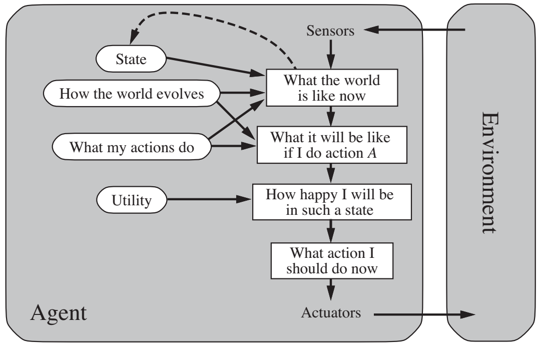
Goal-based agents (cont.)

Remark

The 'global' definition of rationality, designating as rational those agent functions that have highest performance, is turned into a 'local' constraint on rational-agent designs

Goal-based agents (cont.)

A model-based and utility-based agent uses a model along with a utility function that measures its preferences among states of the world and it chooses the action leading to best expected utility


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Utility
How happy I will be in such a state
What action I should do now
Agent
Actuators
Environment

- Expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome

Utility-based agents (cont.)

Is it that simple? We build agents that maximise expected utility?

It's true that such agents would be intelligent, but it's not simple

A utility-based agent has to model and keep track of environment, these very tasks have involved a deal of research on perception, representation, reasoning, and learning

Choosing the utility-maximising course of action is also difficult

- It requires ingenious algorithms and even with these algorithms, perfect rationality is often unachievable in practice because of computational complexity

Learning agents

The agents

Agents and environment
UFC/DC
AI (CK0031)
2016.2

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Learning agents

We described agent programs with various methods for selecting actions but we do not know yet how they come into being

- 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*'

He proposes to build learning machines and then to teach them

Remark

In many areas of AI, this is now the preferred method for creating state-of-the-art systems

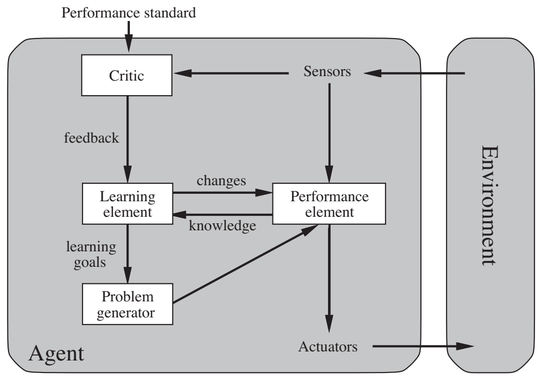
Learning has the advantage that it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow

Agents and environment
UFC/DC
AI (CK0031)
2016.2

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Learning agents (cont.)

A **learning agent** can be divided into four conceptual components



- **learning element**, responsible for making improvements
- **performance element**, responsible for selecting external actions

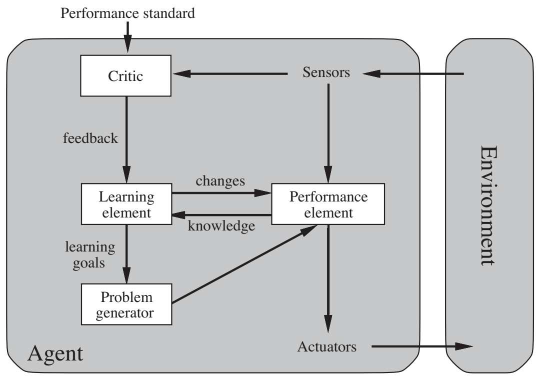
Agents and environment
UFC/DC
AI (CK0031)
2016.2

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Learning agents (cont.)

The performance element is what we considered to be the agent

- It takes in percepts and decides on actions



The learning element uses feedback from the **critic** element on how the agent is doing and determines how the performance element should be modified to do better in the future

Learning agents (cont.)

The design of learning and performance elements depends on the design of the performance element

- When designing an agent that learns a certain capability, the first question is NOT 'How am I going to get it to learn this?'
- We first ask 'What kind of performance element will my agent need to do this once it has learned how?'

Given an agent design, learning mechanisms can be constructed to improve every part of the agent

Learning agents (cont.)

Definition

The critic tells the learning element how well the agent is doing with respect to a fixed performance standard

- The critic is necessary as percepts themselves provide no indication of the agent's success

Example

A chess program could receive a percept indicating that it has checkmated its opponent, but it needs a performance standard to know that this is a good thing, as the percept itself does not say so

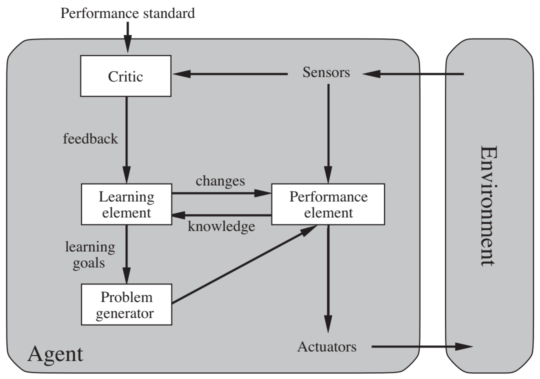
It is important that the performance standard be fixed

- Conceptually, one should think of it as being outside the agent altogether because the agent must not modify it to fit its own behaviour

Learning agents (cont.)

The last component of a learning agent is the **problem generator**

- It is responsible for suggesting actions that will lead to new and informative experiences



Learning agents (cont.)

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 exploratory actions

Learning agents (cont.)

To make the design more concrete, return to the automated taxi

Example

The performance element is whatever collection of knowledge and procedures the taxi has for selecting its driving actions

- The taxi drives using the performance element, the critic observes the world and passes information along to the learning element

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 areas of behaviour in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces and conditions

Learning agents (cont.)

The learning element can make changes to any of the 'knowledge' components in the agent diagrams

- Simplest cases involve learning directly from the percept sequence
- Observation of pairs of successive states of the environment can allow the agent to learn 'How the world evolves', and observation of the results of its actions can allow the agent to learn 'What my actions do'

Example

For example, if the taxi exerts a certain braking pressure when driving on a wet road, then it will soon find out how much deceleration is achieved

Clearly, these two learning tasks are more difficult if the environment is only partially observable

Learning agents (cont.)

These forms of learning do not need to access the external performance standard

- Meaning, the standard is the universal one of making predictions that agree with experiment

The situation is more complex for a utility-based agent that wishes to learn utility information

Example

Suppose the taxi-driving agent receives no tips from passengers who have been thoroughly shaken up during the trip

- The external performance standard must inform the agent that the loss of tips is a negative contribution to its overall performance; then the agent might be able to learn that violent manoeuvres do not contribute to its own utility

Learning agents (cont.)

Remark

The performance standard distinguishes part of the incoming percept as a **reward** (or **penalty**) that provides direct feedback on the quality of the agent's behaviour

- Hard-wired performance standards such as pain and hunger in animals can be understood in this way.

Learning agents (cont.)

In summary, agents have a variety of components that can be represented in many ways within the agent program

- There appears to be variety among learning methods

There is a single unifying theme as learning in intelligent agents is the 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

Agent components
The agents

Agent components

We described agent programs (in high-level terms) as consisting of various components, whose function it is to answer questions

- ‘What is the world like now?’
- ‘What action should I do now?’
- ‘What do my actions do?’

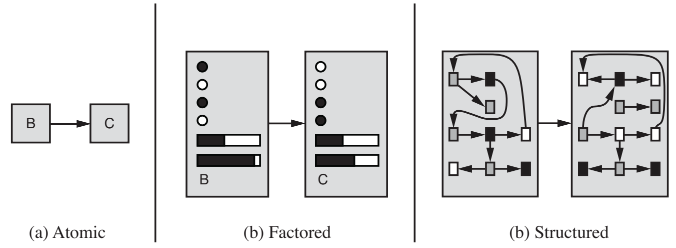
The next question is, ‘How do these components work?’

- Focus on basic distinctions among the various ways that the components can represent the environment that the agent inhabits

Agent components (cont.)

Roughly speaking, we can place the representations along an axis of increasing complexity and expressive power

- **atomic**, **factored**, and **structured**

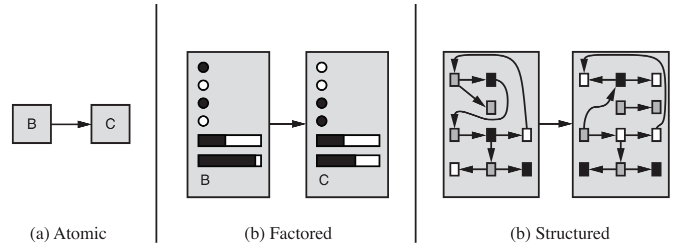


(a) Atomic (b) Factored (b) Structured

Agent components (cont.)

Consider the agent component dealing with 'What my actions do?'

- This component describes the changes that might occur in the environment as the result of taking an action



(a) Atomic (b) Factored (b) Structured

Three ways of schematically depicting states and transitions

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Agent components (cont.)

Definition

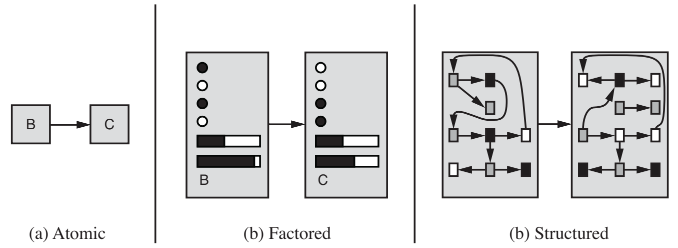
In an **atomic representation** each state of the world is indivisible

- It has no internal structure

Example

Consider the problem of finding a driving route from one end of a country to the other via some sequence of cities

- For the purposes of solving this problem, it may suffice to reduce the state of world to the name of the city we are in

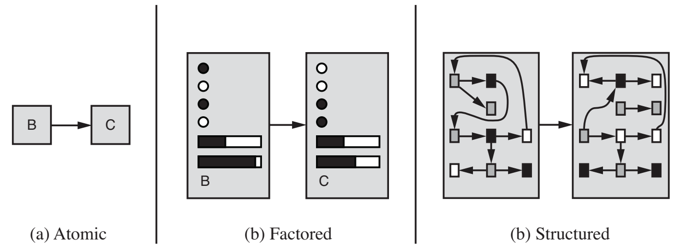


(a) Atomic (b) Factored (b) Structured

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Agent components (cont.)

A single atom, a 'black box' whose only discernible property is that of being identical to or different from another black box



(a) Atomic (b) Factored (b) Structured

Algos underlying search and game-playing, hidden Markov models, and Markov decision processes work with atomic representations

- or, at least, they treat representations as if they were atomic

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Agent components (cont.)

Now consider a higher-fidelity description for the same problem, where we need to be concerned with more than atomic location

Example

- We might need to pay attention to how much gas is in the tank, our current GPS coordinates, whether or not the oil warning light is working, how much spare change we have for toll crossings, what station is on the radio, and so on

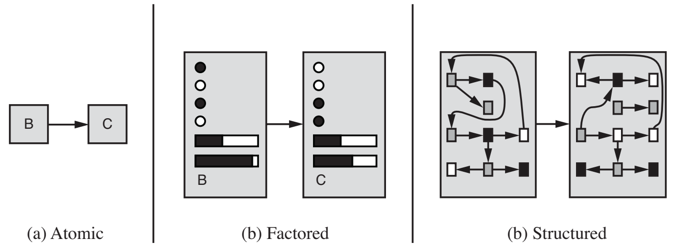
Definition

A **factored representation** splits up each state into a fixed set of variables or attributes, each of which can have a value

Agent components (cont.)

While different atomic states have nothing in common (different black boxes), different factored states can share some attributes (same GPS location) and not others (lots of gas or no gas)

- It is easier to work out how to turn one state into another



(a) Atomic (b) Factored (b) Structured

Agent components (cont.)

With factored representations, we can also represent uncertainty

Many important areas of AI are based on factored representations, including constraint satisfaction algorithms, propositional logic, planning, Bayesian networks, and machine learning algorithms

Agent components (cont.)

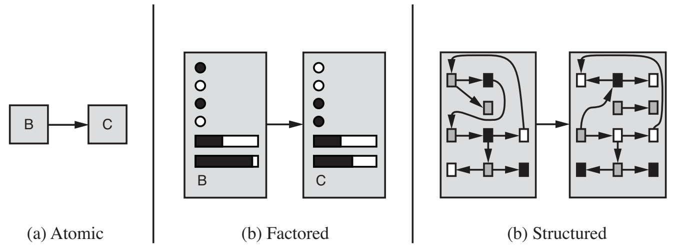
At times, we need to understand the world as having things in it that are related to each other, not just variables with values

Example

- There is a truck ahead of us is reversing into the driveway of a dairy farm but a cow has got loose and is blocking the way
- A factored representation is unlikely to be pre-equipped with a true or false type attribute `Truck-Ahead-Backing-Into-Dairy-Farm-Driveway-Blocked-By-Loose-Cow`

Agent components (cont.)

In a **structured representation**, objects (cows, trucks, ...) and their various and varying relationships can be described explicitly



(a) Atomic (b) Factored (b) Structured

- A state includes objects, each of which may have attributes of its own and/or relations with other objects

Agent components (cont.)

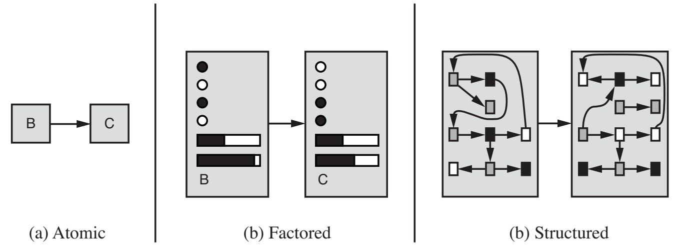
Structured representations underlie relational databases, first-order logic, first-order probability models, knowledge-based learning and much of natural language understanding

- In fact, almost everything that humans express in natural language concerns objects and their relationships

Agent components (cont.)

The axis of atomic ⇒ factored ⇒ structured representations

- the axis of increasing **expressiveness**



(a) Atomic (b) Factored (b) Structured

Roughly, a more expressive representation can capture, at least as concisely, everything a less expressive one can, plus some more

- Often, the more expressive language is much more concise

Agent components (cont.)

Example

- The rules of chess can be written in a page or two of a structured-representation language such as first-order logic
- Thousands of pages when written in a factored-representation language such as propositional logic

Reasoning and learning become more complex as the expressive power of the representation increases

Agent components (cont.)

Remark

To benefit from expressive representations while avoiding drawbacks, intelligent systems for the real world may need to operate at all points along the axis simultaneously

Happy hackin
Agents and environment

Agents and environment
Rationality and good behaviour
Rationality
Omniscience, learning and autonomy
The environment
Specification of task environments
Properties of task environments
The agents
Agent programs
Simple reflex agents
Model-based reflex agents
Goal-based agents
Utility-based agents
Learning agents
Agent components
Happy hackin

Happy hackin

Go to the **code repository of the AIMA book**: `AIMA Code`

- It has implementations of a number of environments, together with a general-purpose environment simulator that places one or more agents in a simulated environment, observes their behaviour over time, and evaluates them according to a given performance measure

Pick code in your favourite language, and play with it!