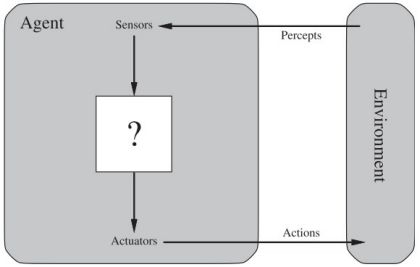
Artificial Intelligence

Agents

Ch 2.1

Agents

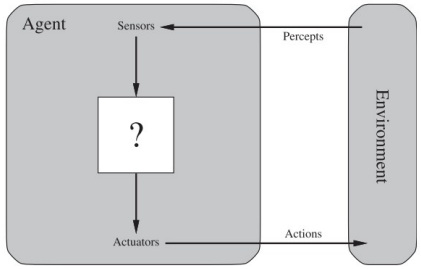
An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting** upon that environment through **actuators**.



- **Percept**: perceptual inputs at any given **instant**.
- **Percept Sequence**: the complete history of everything the agent has ever perceived.

Agents

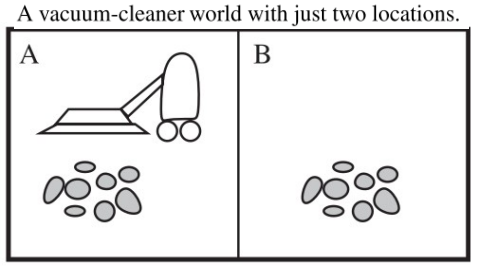
An **agent** is anything that can be viewed as **perceiving** its **environment** through **sensors** and **acting** upon that environment through **actuators**.



- **Percept**: perceptual inputs at any given **instant**.
- **Percept Sequence**: the complete history of everything the agent has ever perceived.

***Important**: an agent's choice of action cannot depend on anything it hasn't perceived*

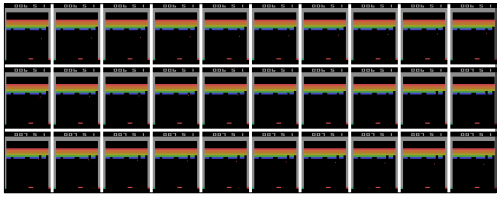
Agents – Example



- **Percept**:
  - square the agent is in
  - whether there is dirt in the square.
- **Actions**:
  - move left,
  - move right,
  - suck up the dirt, or do nothing.

Agents – Example – Game Playing Agent

Percept: A frame of the game



Percept Sequence: The set of all frames observed by the agent.

Actions: press and release actions for all buttons of the joystick.

Agent Function vs Agent Program

- The **agent function** maps any given percept sequence to an action.
- The **agent program** is a concrete implementation, running within some physical system.
- The agent function is an abstract **mathematical** description.

Agent Function

Pure function: no state or side-effects

modification to variables outside the local environment; observable effect besides returning a value

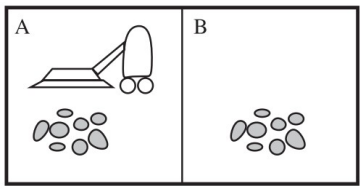
- Cannot store previous percepts.
- The **entire historical percept** sequence has to be passed to the function.

Agent Program

Take the **current percept** as input from the sensors and return an action to the actuators.

nothing more is available from the environment!

The program can **remember** previous percepts (even the entire sequence) if necessary.



A vacuum-cleaner world with just two locations.

Agent Function

Example of Agent Function:

1. if the current square is dirty,
2. then suck;
3. otherwise, move to the other square.

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*

Figure 2.8 The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

Tabulation of the 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*
⋮	⋮

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

For most agents, this would be a **very large** table—infinite, in fact, unless we place a bound on the length of percept sequences we want to consider

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

Agents are Useful "Abstractions"

- The notion of an **agent** is meant to be a **tool for analyzing systems**, not an absolute characterization that divides the world into agents and non-agents.
- A calculator can be viewed as an agent such an analysis would hardly aid our understanding of the calculator.
- AI operates at (what the authors consider to be) the most interesting end of the spectrum, where the artifacts have significant **computational resources** and the task environment requires **nontrivial decision making**.

Artificial Intelligence

Rationality

Ch 2.2

Rational Agents

- Some agents behave better than others.
- A **rational agent** behaves as well as possible **given what it knows**.
  - It does the **"right thing"**.
- Some environments are more difficult than others.
- How well an agent can behave depends on the **nature of the environment.**

How to evaluate if the agent did the right thing?

- Let the agent act in the environment. These actions cause 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.

Important

- Notice 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.

The Performance Measure

- **There is not one fixed performance measure for all tasks and agents**.
- A **designer** will devise one appropriate to the circumstances. This is not as easy as it sounds!

Example
Vacuum-cleaner performance measure

Amount of dirt cleaned up in a single eight-hour shift.

Example
Vacuum-cleaner performance measure

Amount of dirt cleaned up in a single eight-hour shift.

Solution:

- clean up the dirt
- dump it all on the floor
- then clean it up again
- and so on.

More suitable performance measure: reward the agent for having a **clean floor**:

- +1 for each clean square at each time step.
- Penalty for electricity consumed
- Penalty for noise generated.

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

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.

Definition of a **Rational Agent**

For **all** percept sequences, 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.

Example

Vacuum-cleaner:

- If a square is dirty, clean it;
- Else, move to the other square.

Is this rational agent?

Example

Vacuum-cleaner:

- If a square is dirty, clean it;
- Else, move to the other square.

Is this rational agent?

It depends!

- Performance measure?
- Prior knowledge?
- Sensors?
- Actuators?

Example

Vacuum-cleaner:

- If a square is dirty, clean it;
- Else, move to the other square.

Is this rational agent?

- **Performance measure**: +1 for each clean square at each time step, over a "lifetime" of 1000 time steps.
- **Prior**: The "geography" of the environment is known; dirt distribution and the initial location of the agent are not. Clean squares stay clean; sucking cleans the current square. The Left and Right actions move the agent left and right except in the borders.
- **Actions**: Left, Right, and Suck.
- **Perception**: The agent correctly perceives its location and whether that location contains dirt.

Example

Vacuum-cleaner:

- If a square is dirty, clean it;
- Else, move to the other square.

Is this rational agent?

Under these circumstances the agent is rational.

- **Performance measure**: +1 for each clean square at each time step, over a "lifetime" of 1000 time steps.
- **Prior**: The "geography" of the environment is known; dirt distribution and the initial location of the agent are not. Clean squares stay clean; sucking cleans the current square. The Left and Right actions move the agent left and right except in the borders.
- **Actions**: Left, Right, and Suck.
- **Perception**: The agent correctly perceives its location and whether that location contains dirt.

Example

Vacuum-cleaner:

- If a square is dirty, clean it;
- Else, move to the other square.

Is this rational agent?

Under these circumstances
the agent is rational.

What if the performance included a -1 penalty for each movement?

- **Performance measure**: +1 for each clean square at each time step, over a "lifetime" of 1000 time steps.
- **Prior**: The "geography" of the environment is known; dirt distribution and the initial location of the agent are not. Clean squares stay clean; sucking cleans the current square. The Left and Right actions move the agent left and right except in the borders.
- **Actions**: Left, Right, and Suck.
- **Perception**: The agent correctly perceives its location and whether that location contains dirt.

Rationality vs Omniscience

- An omniscient agent knows the actual outcome of its actions and can act accordingly. Omniscience is impossible in reality.
- Rationality is not the same as perfection. Rationality maximizes expected performance, while perfection maximizes actual performance.
- Example: A person cross the street and it is hit by a meteor. Was the person dumb because it crossed the street?
- Important: if an agent does not look both ways before crossing a busy road, then it is being irrational because it is acting with an uninformative percept sequence.
- Information gathering and exploration are important activities.

Learning

- Agents with little or no experience would have to **act randomly**.
- The designer can provide some initial knowledge (**built-in reflexes**) as well as an ability to learn.
- This prior knowledge may be modified and augmented as the agent gains experience.
- After sufficient experience of its environment, the behaviour of a rational agent can become **independent of its prior knowledge**.
- In order to be rational, the agent has to **learn** as much as possible.

Learning and Autonomy

- Example: We can start a game assuming a dice is fair, but observing the outcome of the rolls, we may learn that it is biased towards some numbers.
- Example: A vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

Artificial Intelligence

Environments

Ch 2.3

task environments, which are essentially the "problems" to which rational agents are the "solutions."

Definition of the Task Environment: PEAS

(Performance, Environment, Actuators, Sensors)

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

Figure 2.4 PEAS description of the task environment for an automated taxi.

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

Figure 2.5 Examples of agent types and their PEAS descriptions.

Properties of Task Environments
Fully Observable vs. Partially Observable vs. Unobservable

- **Fully Observable**: agent's sensors give it access to the complete state of the environment at each point in time. A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- **Partially Observable**: 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.
- **Unobservable**: If the agent has no sensors at all then the environment is unobservable.

Properties of Task Environments
Single Agent vs. Multiagent

- An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas two agents cooperating to get an item is a multiagent environment.
- Multiagent environments can be **competitive or cooperative**.
- **Communication often emerges** as a rational behavior in multiagent environments.
- In some competitive environments, **randomized behavior is rational** (for instance, to prevent the enemy from predicting the agent's actions).

Properties of Task Environments
Deterministic vs. Stochastic

- **Deterministic**: next state is completely determined by the current state and the action executed by the agent.
- We say an environment is **uncertain** if it is not fully observable or not deterministic.
- **Stochastic and Nondeterministic Environments**: in nondeterministic environment, the same action can lead to different outcomes but no probabilities are attached to them. In stochastic environments, probabilities are attached.

Properties of Task Environments
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.
- In **sequential environments**, on the other hand, the current decision could affect all future decisions. Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences.

OBS: In reinforcement learning, episodic environments are those in which the agent can try to solve the task several times performing as many actions as the environment allows (as if it could experience life several times until it figures out how to have a successful life), while in sequential environments, the agent has a single opportunity to try to solve the task.

Properties of Task Environments

- **Static vs. Dynamic**: In dynamic tasks, the environment can change while an agent is deliberating (deciding what to do).
- **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.
- **Known vs. unknown**: In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given. Solitaire card games: known environment (rules) and partially observable. A new video game can be unknown (what the buttons do?) and fully observable (the screen may show the entire game state).

The hardest tasks:

- **partially observable**
- **multiagent**
- **stochastic**
- **sequential**
- **dynamic**
- **continuous**
- and **unknown**.

Taxi driving a rented car in a new country with unfamiliar geography and traffic laws.

Figure 2.6 Examples of task environments and their characteristics.

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

Note: Several of the answers in the table depend on how the task environment is defined.
Example: What about patients, staff, and the pharmaceutical industry in medical diagnosis?

Artificial Intelligence

Types of Agents

Ch 2.4

The Structure of Agents

- Once the task environment is defined, we can design rational agents to solve the tasks.
- The job of AI is to design an **agent program** that implements the **agent function**—the mapping from percepts to actions.
- We assume this program will run on some sort of **computing device** with **physical sensors and actuators**—we call this **the architecture**.

Types of Agents

- **Simple reflex agents**;
- **Model-based agents**;
- **Goal-based agents**;
- **Utility-based agents;**
- **Learning agents.**

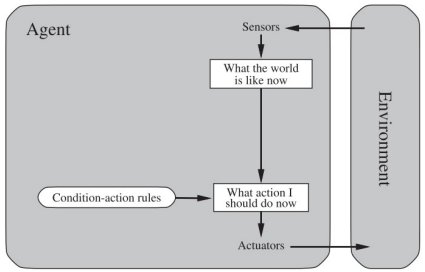
Simple Reflex Agents

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

Example

```
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
```

Example: 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. In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking." Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a condition–action rule (also called situation–action rules, productions, or if–then rules).



Schematic diagram of a simple and general reflex agent. rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process

Simple Reflex Agents

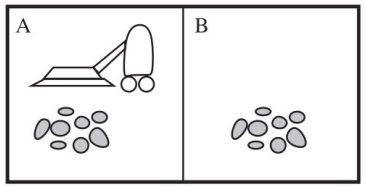
```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

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

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

Critic to Simple Reflex Agents

They work only if the environment is **fully observable**.



Example: only a dirty sensor with two possible values: [Dirty] and [Clean].

It can Suck in response to [Dirty];
what should it do in response to [Clean]?

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.
- Example: For driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once.

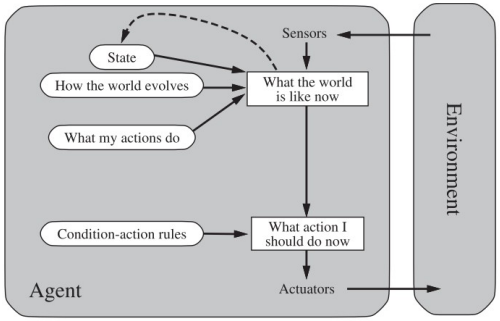
The Model of the World

- Updating this internal state information requires knowing:
  - how the world evolves independently of the agent.
  - how the agent's own actions affect the world.
- This knowledge about "**how the world works**"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model of the world**. An agent that uses such a model is called a **model-based agent**.

An overtaking car generally will be closer behind than it was a moment ago.

Ex1: When the agent turns the steering wheel clockwise, the car turns to the right.
Ex2: 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



Model-based Reflex Agents

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*

The function UPDATE-STATE is responsible for creating the new internal state description. The details of how models and states are represented vary widely depending on the type of environment and the particular technology used in the agent design. We will see some approaches in the course.

Model-based Reflex Agents

- Regardless of the kind of representation used, it is seldom possible for the agent to determine the current state of a partially observable environment exactly. Instead, the box labeled "what the world is like now" represents the agent's "**best guess**" (or sometimes best guesses).
  - Ex.: 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

Goal-based Agents

- Knowing something about the current state of the environment is not always enough to decide what to do.
- The correct decision depends on where the taxi is trying to get to. In other words, the agent needs some sort of **goal** information that describes **situations that are desirable** - for example, being at some destination point.
- **Search** and **planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals.
- Notice that decision making of this kind is fundamentally different from the condition-action rules described earlier, in that it involves **consideration of the future** - both "What will happen if I do such-and-such?" and "Will that make me happy?"

Utility-based Agents

- Goals alone are not enough to generate high-quality behavior in most environments.
  - **Ex.: many action sequences will get the taxi to its destination, but some are quicker, safer, more reliable, or cheaper than others.**
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent.
- Because "happy" does not sound very scientific, economists and computer scientists use the term **utility** (the quality of being useful) instead.

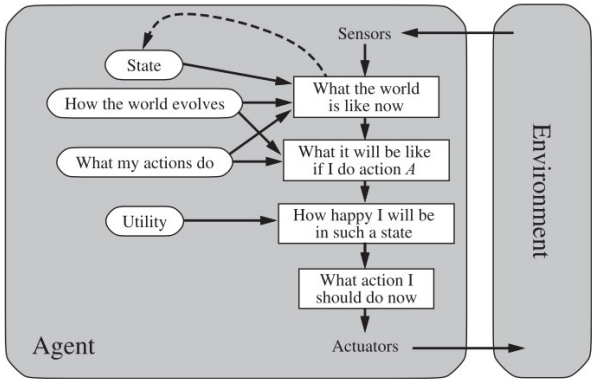
The Utility Function

- We have already seen that a **performance measure** assigns a score to any given sequence of environment states, so it can easily distinguish between more and less desirable sequences of states.
- An agent's **utility function** is essentially an **internalization** of the performance measure.
- If the internal utility function and the external performance measure are in **agreement**, then an agent that chooses actions to maximize its utility will be rational according to the external performance measure.

Utility-based Agents >>> Goal-based Agents

- When there are **conflicting goals**, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff.
- When there are **multiple 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.

A Model-based, Utility-based Agent

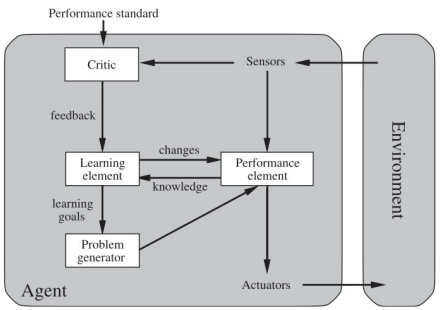


It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

Learning Agents

- We have described agent programs with various methods for selecting actions. We have not, so far, explained how the agent programs **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."
- 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.

Learning Agents



A General Learning Agent

- The **learning element** is responsible for making improvements
- The **performance element** is responsible for selecting external actions.
  - It is what we have previously considered to be the entire agent which 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** suggests actions that will lead to new and informative experiences.
  - Exploratory actions can lead to suboptimal actions in the short run, but the agent might discover much better actions for the long run.

Example: A Learning Taxi Agent

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

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

Artificial Intelligence

State Representation

Ch 2.4.7

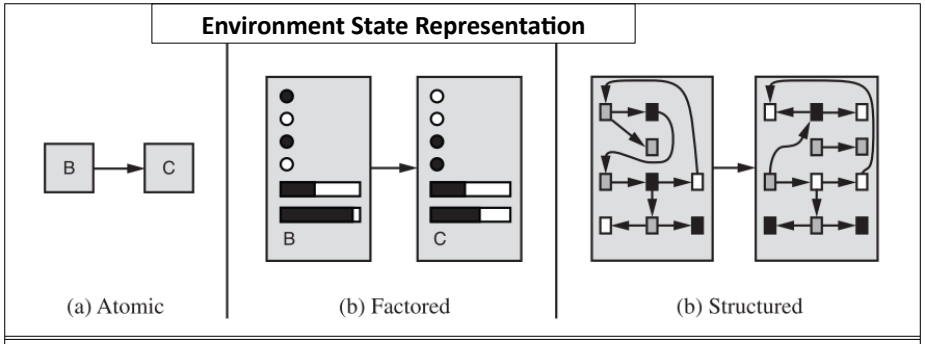


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

Atomic Representations

- 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 just the name of the city we are in—**a single atom of knowledge**; a "black box" whose only discernible property is that of being identical to or different from another black box.

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

Consider a higher-fidelity description for the same problem, where we need to be concerned with **more than just atomic location** in one city or another; we might need to pay attention to:

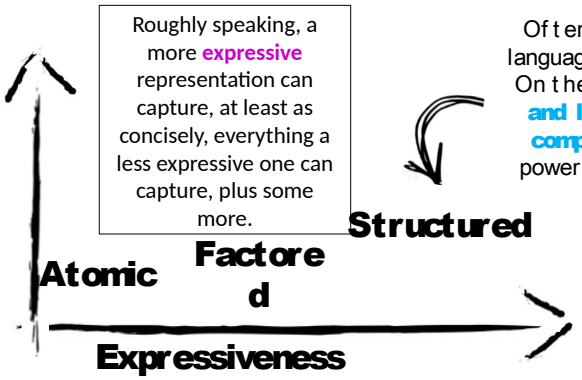
- how much gas is in the tank;
- our current GPS coordinates;
- whether or not the oil warning light is working;
- how much spare change we have for toll crossings;
- what station is on the radio, and so on.

Factored Representations

- While two different atomic states have nothing in common—they are just different black boxes—two different factored states can share some attributes (such as being at some particular GPS location) and not others (such as having lots of gas or having no gas).
- With factored representations, we can also represent **uncertainty**—for example, ignorance about the amount of gas in the tank can be represented by leaving that attribute blank.

- For many purposes, we need to understand the world as having things in it that are **related** to each other, not just variables with values.
- For example, we might notice that a large truck ahead of us is reversing into the driveway of a dairy farm but a cow has got loose and is blocking the truck's path.
- A factored representation is unlikely to be pre-equipped with the **attribute** TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow with **value true or false**.

Instead, we would need a **structured representation**, in which **objects** such as cows and trucks and their various and varying **relationships** can be described explicitly.



To gain the benefits of expressive representations while avoiding their drawbacks, intelligent systems for the real world may need to operate at all points along the axis simultaneously

Summary

- An **agent** is something that perceives and acts in an environment. The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.
- The **performance measure** evaluates the behavior of the agent in an environment. A **rational agent** acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Task environments vary along several significant dimensions. They can be fully or partially observable, single-agent or multiagent, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.
- The **agent program** implements the agent function. There exists a variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept. **Goal-based agents** act to achieve their goals, and **utility-based agents** try to maximize their own expected "happiness."
- All agents can improve their performance through **learning**.