Course: Introduction to AI

Prof. Ahmed Guessoum
The National Higher School of AI

Chapter 2

Intelligent Agents

Outline

- Agents and Environments
- Good Behavior: The Concept of Rationality
  - Rationality
  - Omniscience, learning, and autonomy
- The Nature of Environments
  - Specifying the task environment
  - Properties of task environments
- THE STRUCTURE OF AGENTS
  - Agent programs
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents
  - How the components of agent programs work

Aim of this chapter

- **Rational agents** as central to our approach to artificial intelligence
- The concept of **rationality** can be applied to a wide variety of agents operating in any imaginable environment
- Use this concept to develop a small set of design principles for building successful agents—systems that can reasonably be called **intelligent**.

Agents and Environments: Agents

- **Agent:** anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators. E.g.:**

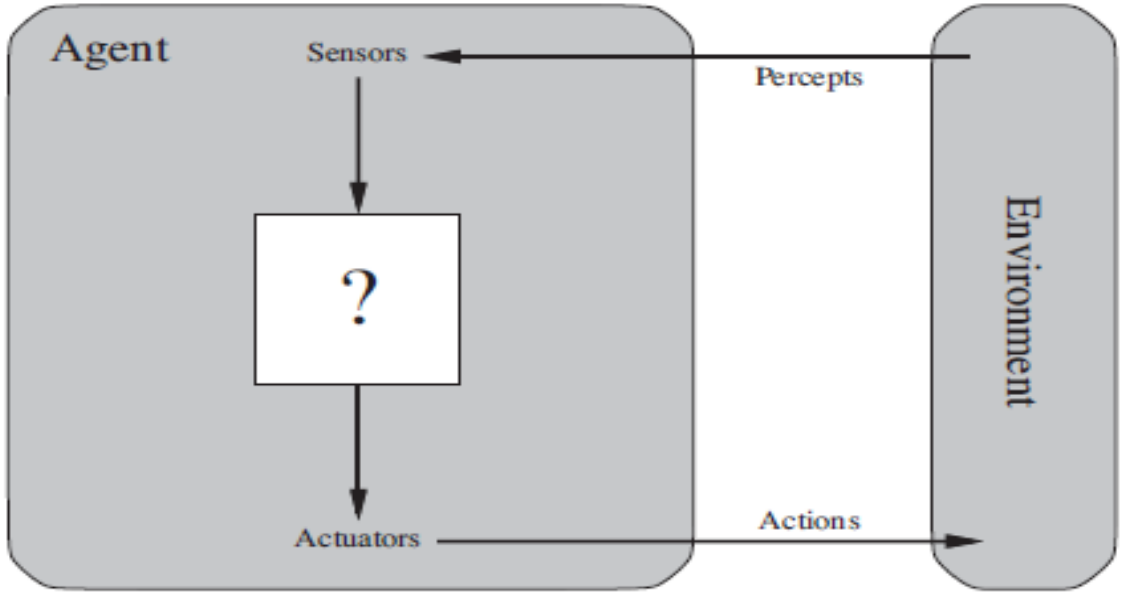


Figure 2.1 Agents interact with environments through sensors and actuators.

Agents and Environments: Agents

Examples of Agents:

- Human agent:
  - Sensors: eyes, ears, hands, …
  - Actuators: hands, legs, vocal tract, …
- Robotic agent:
  - Sensors: cameras, infrared and sonar range finders…
  - Actuators: various motors, robot arm...
- Software agent:
  - Sensory inputs: keystrokes, file contents, network packets, …
  - Actions: displaying on the screen, writing files, sending network packets, …

Agents and Environments: Agents

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

Agents and Environments: Agents

- The **agent function** can be pictured as a table of mappings between percept sequences and actions.
- Such a table is an external (abstract mathematical) characterization of the agent function.
- *Internally*, the agent function for an artificial agent will be implemented within some physical system by an **agent program**.

Agent example

- Vacuum-cleaner world
  - 2 locations A and B
  - Agent **perceives** which square it is in and whether there is dirt in the square
  - Actions: *move left*, *move right*, *suck up the dirt*, or *do nothing.*

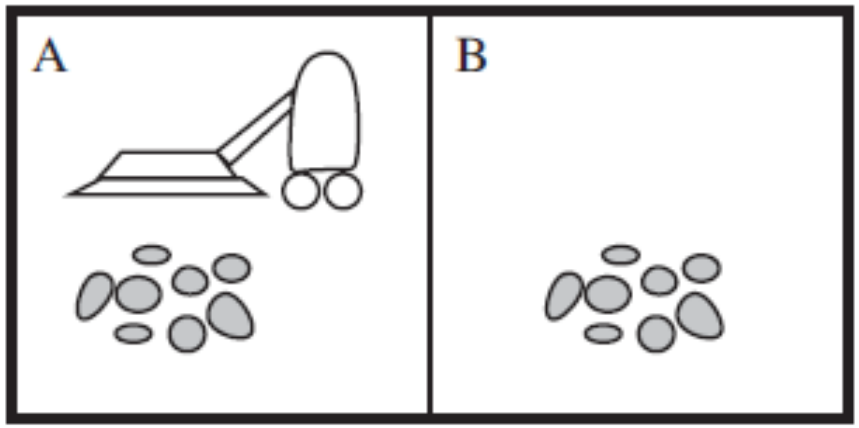


Figure 2.2 A vacuum-cleaner world with just two locations.

Partial tabulation of vacuum-cleaner agent function

Percept sequence Action	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
.	.
.	.
.	.

Agent function design

- So there is a need to define the *Actions* corresponding to the percepts.
- **Question***:*
  - *What is the right way to fill out the table? (*i.e. what makes an agent good or bad, intelligent or stupid?)
- **N.B.:**
  - All areas of engineering can be seen as designing artifacts that interact with the world; but
  - **AI** operates where the artifacts have **significant computational resources** and the task environment requires **nontrivial decision making.**

Outline

- Agents and Environments
- Good Behavior: The Concept of Rationality
  - Rationality
  - Omniscience, learning, and autonomy
- The Nature of Environments
  - Specifying the task environment
  - Properties of task environments
- THE STRUCTURE OF AGENTS
  - Agent programs
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents
  - How the components of agent programs work

GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

- A **rational agent** is one that does the *right thing.* (i.e. every action in the table is *correct*)
- What does "right thing" ("correct action") mean?
- Obvious approach: consider the *consequences* of the agent's behaviour.

  Precepts → sequences of actions →

  environment goes through a sequence of states
- Desirability is captured by a **performance measure** that evaluates any given sequence of environment states.
- Performance measure is designed specifically for the task at stake.

Performance measure for vacuum cleaner agent

- Amount of dirt cleaned in a fixed period of time?
  - *Risk: agent cleans up the floor well, dumps the dirt, cleans up again, etc. ➔ not a good performance measure*
- Alternative: reward the agent for having a clean floor. ➔ it is a more suitable measure
  - Award one point for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated).
- *General rule: design performance measures according to expectations in the environment.*
- *N.B.:* "*Average* cleanliness over time" *is questionable!*

Rationality

- What is rational at any given time depends on four factors:
  - 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 each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rationality: vacuum-cleaner agent

- if square is dirty clean it; else move to the other square. (function previously tabulated)
- Is this a rational agent? (Yes; left as textbook exercise.)
- We need to analyse the enunciated 4 factors.
  - Performance measure: award one point for each clean square at each time step, over 1000 time steps period.
  - Only available actions: Left, Right, and Suck.
  - Environment "geography" known a priori; but not dirt distribution nor agent initial location. Clean squares stay clean; sucking cleans current square. Left and Right actions move agent left and right except when this would take agent outside the environment, in which case agent remains where it is.
  - The agent correctly perceives its location and whether that location contains dirt.

Rationality: vacuum-cleaner agent

- Note that 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 perform poorly. A better agent for this case would do nothing once it is sure that all the squares are clean.
  - If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed.
  - If the geography of the environment is unknown, the agent will need to explore it rather than remain in square A or B.
- (Exercise 2.2 asks you to design agents for these cases.)

Omniscience, learning, & autonomy

- **Omniscient agent**: agent who knows the *actual* outcome of its actions and can act accordingly;
- Omniscience is impossible in reality; we cannot anticipate everything.
- Rationality is not the same as perfection; rationality maximizes *expected* performance, while perfection maximizes *actual* performance. The latter is impossible.
- Doing actions *in order to modify future percepts*— sometimes called **information gathering**— is an important part of rationality.
  - Example, it would not be rational to cross the road given an uninformative percept sequence (e.g. look right only)

Omniscience, learning, & autonomy

- Our definition requires a rational agent not only to gather information but also to **learn** as much as possible from what it perceives.
- The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented.
- If an agent relies on the prior knowledge of its designer rather than on its own percepts, the agent is said to lack **autonomy**.
- A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge.
  - E.g. vacuum cleaner learning where to foresee dirt better than one that does not.

Omniscience, learning, & autonomy

- Complete autonomy from the start is seldom required: with little or no experience, the agent would have to act randomly unless the designer gave some assistance.
- 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 behavior of a rational agent can become effectively ***independent*** of its prior knowledge.

Outline

- Agents and Environments
- Good Behavior: The Concept of Rationality
  - Rationality
  - Omniscience, learning, and autonomy
- The Nature of Environments
  - Specifying the task environment
  - Properties of task environments
- THE STRUCTURE OF AGENTS
  - Agent programs
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents
  - How the components of agent programs work

Specifying the task environment

- In designing an agent, the first step must always be to specify the task environment as fully as possible.
- **Task environment:** **P**erformance, **E**nvironment, **A**ctuators, **S**ensors (**PEAS**).
- Example: PEAS description of the task environment for an automated taxi.
- Agent Type: Taxi Driver:
  - **P**erformance Measure: Safe, fast, legal, comfortable trip, maximize profits, …
  - **E**nvironment: Roads, other traffic, pedestrians, customers
  - **A**ctuators: Steering, accelerator, brake, signal, horn, display
  - **S**ensors: Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard/microphone

PEAS for additional agent types

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

- The range of task environments that might arise in AI is vast.
- We can identify a fairly small number of dimensions along which task environments can be categorized.

Fully observable vs. partially observable

- Do the agent's sensors give it access to the complete state of the environment?
  - For any given world state, are the values of all the variables known to the agent?

vs.

Source: L. Zettlemoyer

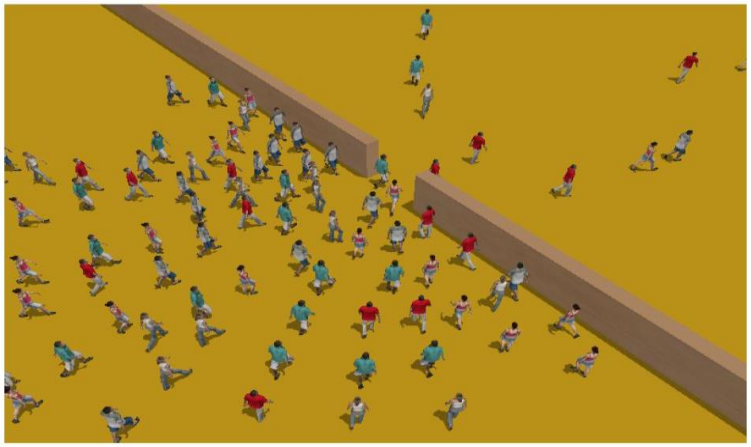
Fully observable vs. partially observable

- 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.
- Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- An environment might be **partially observable** because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data.
  - a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares;
  - an automated taxi cannot see what other drivers are thinking.
- Agent has no sensors ➔ environment is **unobservable**

Single-agent vs. multiagent

- Is an agent operating by itself in the environment?
- Examples:
  - an agent solving a crossword puzzle by itself is clearly in a single-agent environment,
  - an agent playing chess is in a two-agent environment.

vs.



Agent or object?

- Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated just as an object behaving according to the laws of physics?
- **key distinction**: Is B's behaviour best described as maximizing a performance measure whose value depends on agent A's behavior?
  - E.g., in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a **competitive** multiagent environment.
  - In taxi-driving environment, avoiding collisions maximizes the performance measure of all agents, so it is a **partially cooperative** multiagent environment. Also **partially competitive** because, for example, only one car can occupy a parking space.

Deterministic vs. stochastic

- Is the next state of the environment completely determined by the **current state** and the **agent's action**?
  - Is the transition model **deterministic** (unique successor state given current state and action) or **stochastic** (distribution over successor states given current state and action)?
  - **strategic:** the environment is deterministic except for the actions of other agents

vs.

Deterministic vs. stochastic

- Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic.
- Taxi driving is clearly stochastic in this sense, because
  - One can never exactly predict the behaviour of traffic;
  - Also, one's tires may blow out and one's engine may fail without warning.
- The vacuum world as described is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism.

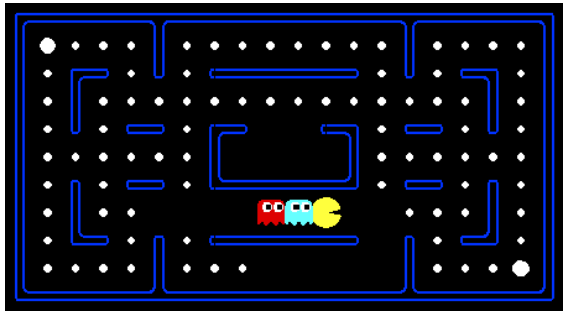
Deterministic vs. stochastic

- We say an environment is **uncertain** if it is not fully observable or not deterministic.
- Our use of the word "**stochastic**" generally implies that uncertainty about outcomes is quantified in terms of probabilities.
- A **nondeterministic** environment is one in which actions are characterized by their *possible* outcomes, but no probabilities are attached to them.
- Nondeterministic environment descriptions are usually associated with performance measures that require the agent to succeed for *all possible* outcomes of its actions.

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
- The next episode does not depend on the actions taken in previous episodes.
- E.g. Classification tasks such as spam detection

vs.

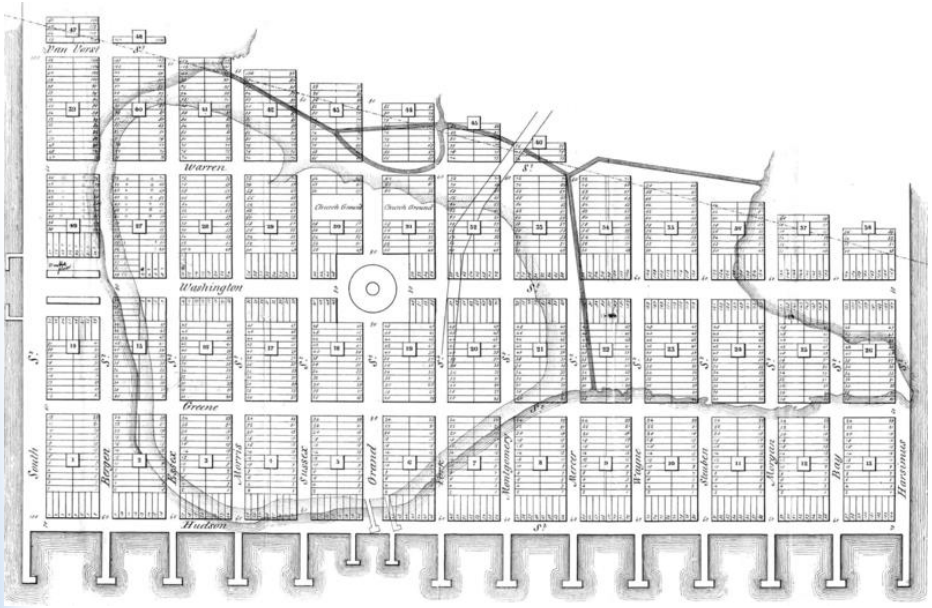


Episodic vs. sequential

- In **sequential** environments, 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.
- Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

Static vs. dynamic

- Is the world changing while the agent is thinking?
  - If so **dynamic**, else **static**.



vs

.

GFDL Image by Minesweeper, 2005

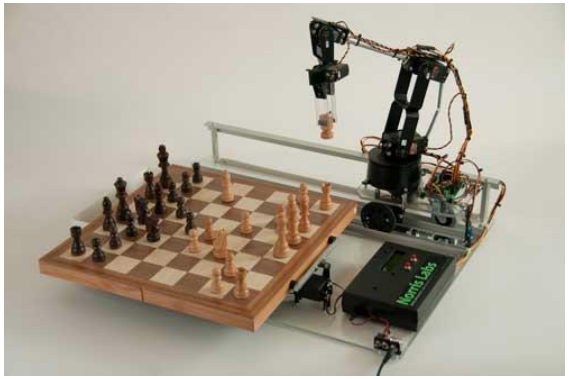
Static vs. dynamic

- If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semi-dynamic**.
  - Taxi driving is dynamic: the other cars and the taxi itself keep moving while the driving algorithm delibers about what to do next.
  - Chess, when played with a clock, is semi-dynamic.
  - Crossword puzzles are static.

Discrete vs. continuous

- The discrete/continuous distinction applies to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent.
- The chess environment has a finite number of distinct states. Chess also has a discrete set of percepts and actions.

vs.



Discrete vs. continuous

- Taxi driving is a continuous-state and continuous-time problem:
  - The speed and location of the taxi and of the other vehicles go through a range of continuous values and do so smoothly over time.
  - Taxi-driving actions are continuous (steering angles, etc.).
- Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

Known vs. unknown

- Are the rules of the environment (transition model and rewards associated with states) known to the agent?
  - Strictly speaking, this is not a property of the environment, but of the agent's state of knowledge about the "laws of physics" of env.

Monopoly, Parker Brothers 1935; photo CC-BY-NC Fir0002/Flagstaffotos

vs.

Myst, © Cyan Worlds, 1993

Known vs. unknown

- In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given.
- Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions.
- Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments.
- It is quite possible for a *known* environment to be *partially* observable—
  - E.g., in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over.
- Conversely, an *unknown* environment can be *fully* observable
  - In a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

Examples of Environments

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

Figure 2.6 Examples of task environments and their characteristics.

Outline

- Agents and Environments
- Good Behavior: The Concept of Rationality
  - Rationality
  - Omniscience, learning, and autonomy
- The Nature of Environments
  - Specifying the task environment
  - Properties of task environments
- THE STRUCTURE OF AGENTS
  - Agent programs
  - Simple reflex agents
  - Model-based reflex agents
  - Goal-based agents
  - Utility-based agents
  - Learning agents
  - How the components of agent programs work

The Structure of Agents

- How does an agent work? How do we implement its behavior?
- **AI's aim**: design an **agent program** that maps percept sequences into actions.
- Program assumed to run on some computing device with physical sensors and actuators, called the **architecture**:

  agent = *architecture* + *program*

- The program must take into account the architecture (PC, robot, etc.) in the commands it issues: e.g. *walk*, *move*, …

Agent programs

- We could think of a lookup of a percept sequence into the *table of actions* ➔ Need to select the appropriate action for every possible percept sequence.

```
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
```

Figure 2.7 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

- The table-driven approach to agent design is doomed to failure.
- Let P be the set of possible percepts and T the lifetime of agent (total number of percepts it will receive), then lookup table will contain $\sum_{t=1}^{T} |P|^t$ entries ➔ **Complexity!**
- Potentially huge space complexity!
  - Autonomous taxi: visual input from a single camera comes in at the rate of ~ 27 megabytes per second (30 frames per second, 640×480 pixels with 24 bits of color information). This gives a lookup table with over $10^{250,000,000,000}$ entries for an hour's drive.
  - Lookup table for chess—a tiny, well-behaved fragment of the real world—would have at least 10^{150} entries
- ➔ we will consider simple agent programs that take a *percept* (not a percept sequence) as input from the sensors and produce an *action* to the actuators.

Agent programs

- Such tables are too huge in terms of space complexity and impossible to be comprehensive.
- Key challenge of AI: build programmes that produce rational behavior from a fairly small program rather than from a vast table.
- There are basic kinds of agent programs that embody the principles underlying almost all intelligent systems. I
- Introduced now…

Simple reflex agents

- **Simple reflex agents**: the simplest type of agents; select actions on the basis of the *current* percept, ignoring the rest of the percept history.
- E.g. Vacuum agent (with table seen before).

```
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.

- Reduction from 4^T possibilities to 4 and when square is *dirty* same action (indep. of location)

Simple reflex agents

- Simple reflex behaviors occur even in more complex environments.
- Some percepts can trigger some established connection in the agent program to some action.
- Such a connection is called **condition–action rule**, e.g.

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

- These reflexes are common for humans, whether learned or "natural" (such as blinking when something approaches the eye).

Simple reflex agents

- A more general and flexible approach is first to build a **general-purpose interpreter for condition–action rules** and **then** to create **rule sets for specific task environments**.

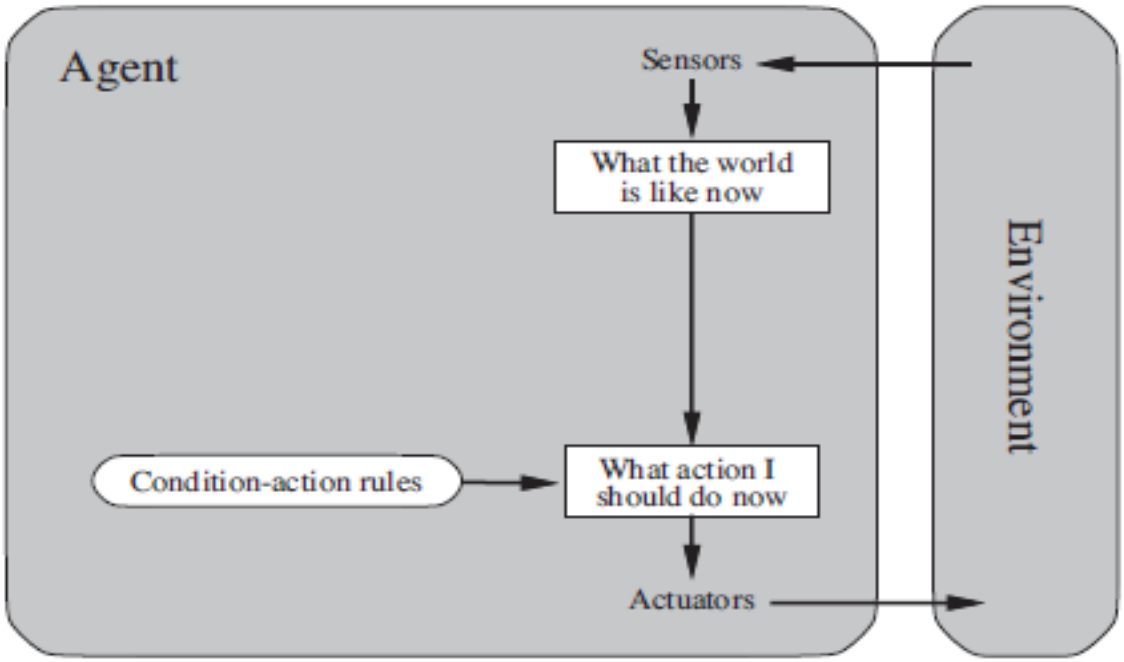


Figure 2.9 Schematic diagram of a simple reflex agent.

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
```

Figure 2.10 A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

- **N.B.:** the description in terms of "rules" and "matching" is purely conceptual; actual implementations can be as simple as a collection of logic gates implementing a Boolean circuit.

Simple reflex agents

- Simple reflex agents:
  - have the admirable property of being simple, **BUT**
  - are of limited intelligence
  - will work *only if the correct decision can be made on the basis of only the current percept, i.e. only if the environment is fully observable.* ➔ Even little unobservability can cause serious trouble.
  - Example: it is not always possible to tell from a single image whether the car is braking ➔ simple percept (video frame) not enough!

Simple reflex agents

- In partially observable environments, infinite loops are often unavoidable for simple reflex agents.
  - E.g. If no sensing of location and square clean, vacuum might attempt *left* infinitely.
- A solution to escape from infinite loops is if the agent can **randomize** its actions.
  - E.g.: flip a coin to choose between *left* and *right*
- A randomized simple reflex agent might outperform a deterministic simple reflex agent.
- Randomized behavior of the right kind can be rational in some *multiagent* environments. In *single-agent* environments, randomization is usually *not* rational.

Model-based reflex agents

- For the agent to handle partial observability it needs to *keep track of the part of the world it can't see now.*

➔ 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. E.g.

  - For braking, keep previous frame for braking front car.
  - For driving task like changing lanes, keep track of other cars positions.
  - *Etc.*

Model-based reflex agents

- To update the internal state information as time goes by (UPDATE-STATE in pseudo-code), two kinds of knowledge need to be encoded:
  1. Some information about how the world evolves independently of the agent.
     - E.g. consequences of a car overtaking another
  2. Some information about how the agent's own actions affect the world.
     - E.g. consequences of turning steering wheel left or right, of driving northward, etc.
- This is called a **model** of the world.
- An agent that uses such a model is called a **model-based agent**.

Model-based reflex agents

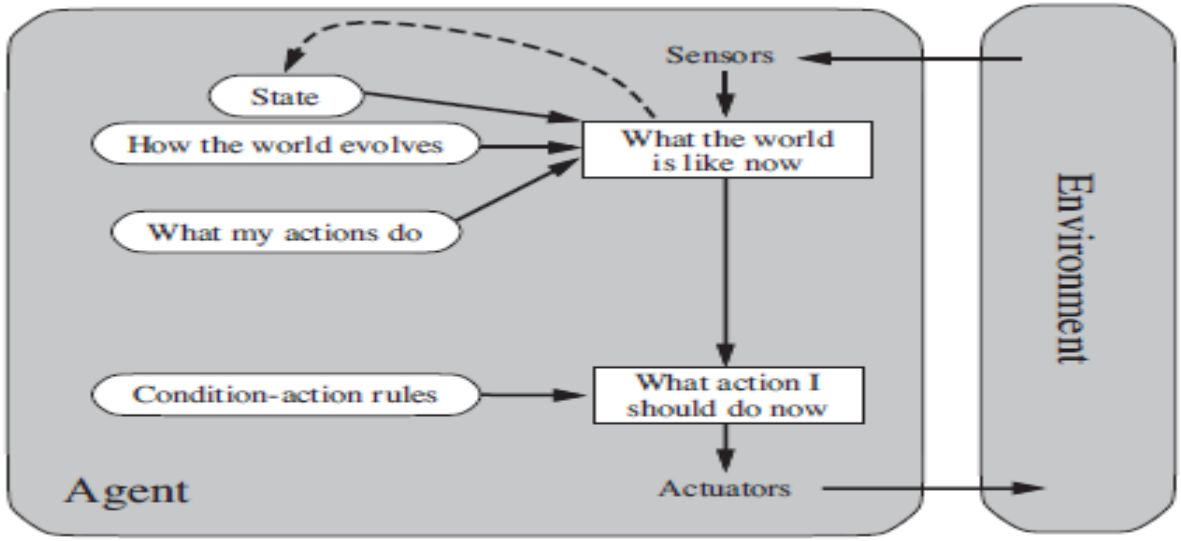


Figure 2.11 A model-based reflex agent.

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*

Figure 2.12 A model-based reflex agent. It keeps track of the current state of the world, using an internal model. It then chooses an action in the same way as the reflex agent.

Model-based reflex agents

- The details of how models and states are represented vary a lot depending on the type of environment and the particular technology used in the agent design.
- Independently of the representation used
  - An agent can usually not determine the current state of a partially observable environment *exactly*;
  - "**best guess**" (or guesses) **to represent the current state**. E.g. best guess of what is blocking a large truck in front of the agent-car?

Goal-based agents

- Knowing something about the current state of the environment is not always enough to decide what to do.
  - E.g.: **at next crossing**, take left, right or go straight?
- Besides a current state description, the agent needs some sort of **goal** information that describes **situations that are desirable**.
  - E.g. being at the passenger's destination
- The goal information can be combined with the model as seen in model-based reflex agents.

Goal-based agents

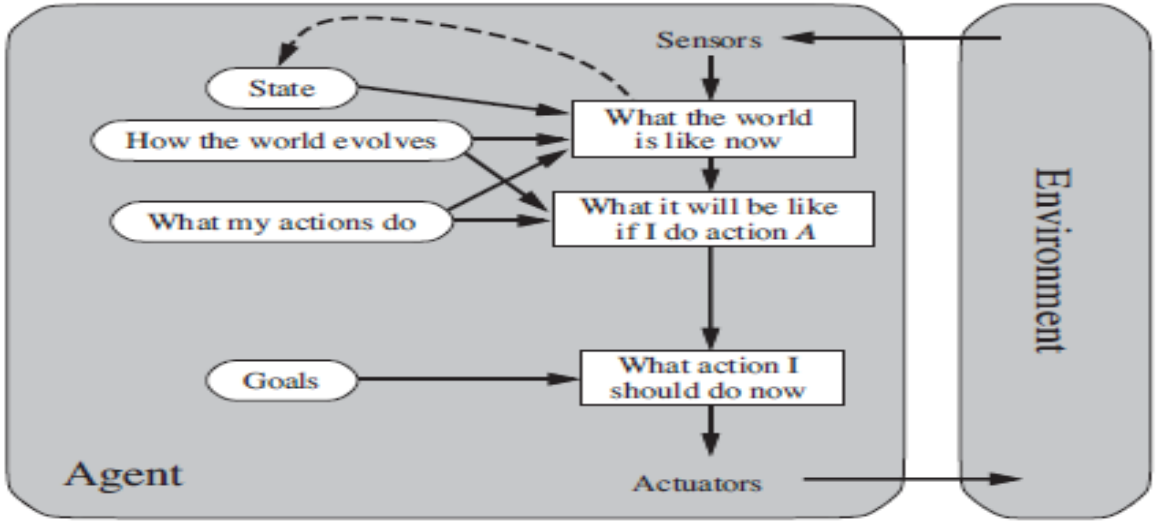


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

- Sometimes goal-based action selection is straightforward
  - Selected action leads to goal satisfaction
- Often, there is a need for a long sequence of actions to achieve the goal (sequence of sub-goals)
  - The Search and Planning subfields of AI
- A goal-based agent may be less efficient, **BUT** it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified. (e.g. different destination)

Utility-based agents

- Goals are not enough to get high-quality agent behavior. E.g.:
  - Various action sequences to reach a destination: some are quicker, safer, more reliable, or cheaper than others.
- It is not "goal achieved or not?", but rather **how satisfied the agent is with the achievement of the goal**.
  ➔ So need for a more general performance measure to compare various world states: **utility**.
- An agent's **utility function** is an internalization of the performance measure.
- **If** internal utility function and external performance measure are in agreement, then an agent that maximizes its utility will be **rational** wrt external performance measure.

Utility-based agents

- Utility-based agent has many advantages in terms of flexibility and learning.
- In two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions.
  1. When there are conflicting goals, only some of which can be achieved. E.g. speed and safety,
     ➔ the utility function specifies the appropriate tradeoff.
  2. When agent can aim for several goals, 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 rational utility-based agent chooses the action that maximizes the **expected utility** of the action i.e., the utility it expects to derive, on average, given the probabilities and utilities of each outcome.

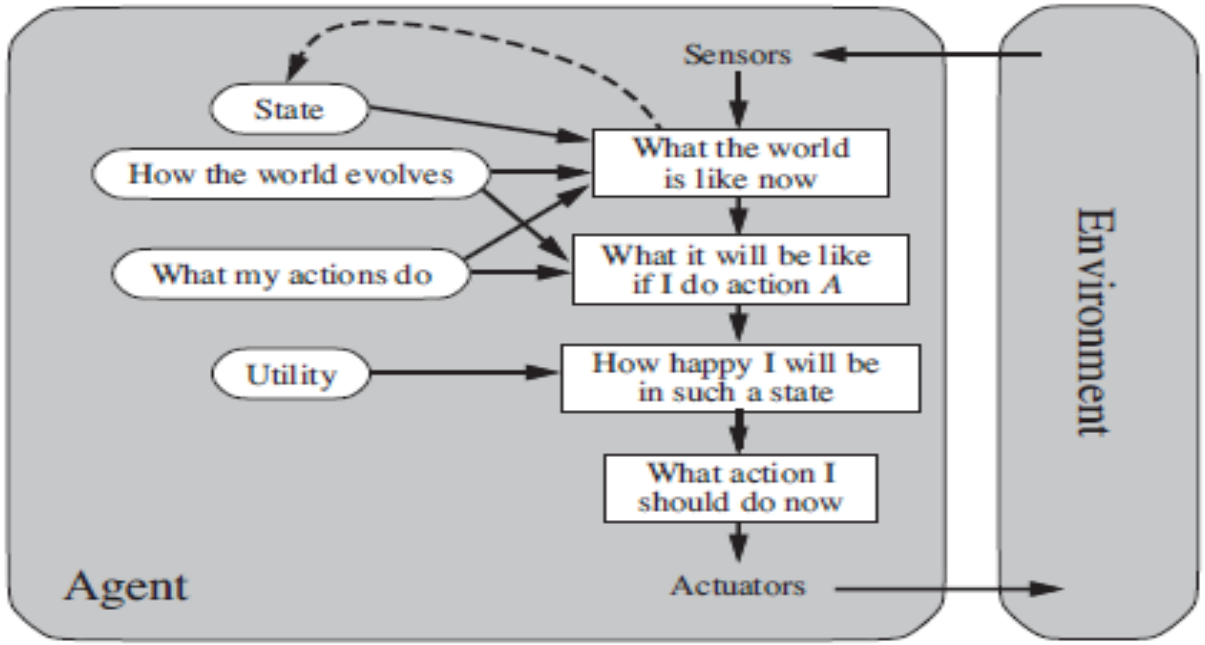


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

- *Any rational agent must behave as if it possesses a utility function whose expected value it tries to maximize.*
- So just build agents that maximize expected utility?
- Yes; but to model and keep track of the environment, a great deal of research on perception, representation, reasoning, & learning has been done.

Learning agents

- Turing had already suggested building *learning machines* and *teach them*.
- In many areas of AI, this is now the preferred method for creating state-of-the-art systems.
- Learning agent can be seen as having 4 components:

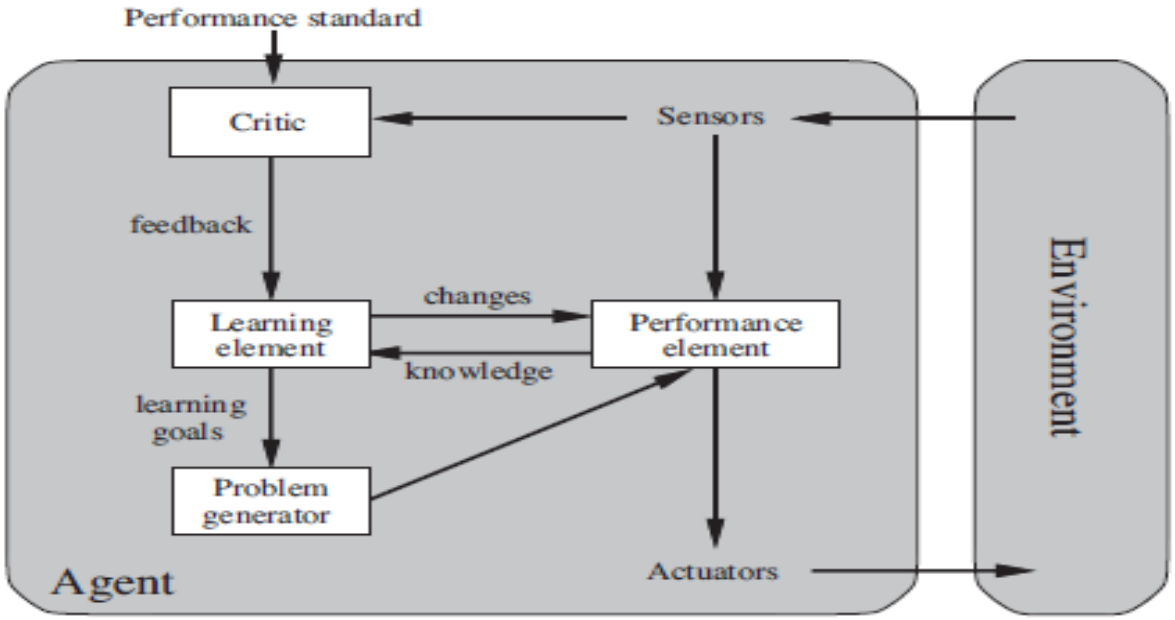


Figure 2.15 A general learning agent.

Learning agents

- **Learning element**: responsible for making improvements of agent (by modifying its "knowledge")
- **Performance element**: responsible for selecting external actions; i.e. takes in percepts and decides on actions
- The learning element uses feedback from the **Critic** on how well the agent is doing with respect to a fixed performance standard and determines how the performance element should be modified to do better in the future.
- The critic is necessary because the percepts themselves provide no indication of the agent's success. E.g.
  - In chess: a percept that agent checkmated opponent is interpreted by critic as good based on a fixed (external) performance standard; percept does not say so.

Learning agents

- **Problem generator**: responsible for suggesting actions that will lead to new and informative experiences ➔
  - agent's exploration of some perhaps suboptimal actions in the short run might lead to discovery of much better actions for the long run.
  - The generator's job is to suggest these exploratory actions.

Example: Automated Taxi

- The **performance element**: collection of knowledge and procedures the taxi has for selecting its driving actions.
- The **critic** observes the world and passes information along to the learning element.
  - E.g. critic observes rude language used by other drivers on agent's rough move (crossing lanes abruptly)
- The **learning element** is able to formulate a rule saying this was a bad action.
- The **performance element** is modified by installation of the new rule. (Ensuring the consistency of the rules!)
- The **problem generator** might identify certain areas of behavior in need of improvement and suggest experiments.

Learning agents: learning element

- The learning element can make changes to any of the "knowledge" components shown in the agent diagram.
  - Learning from percept sequences.
  - Observation of pairs of successive states of the environment ➔ the agent can learn "How the world evolves".
  - Observation of the results of its actions can allow the agent to learn "What my actions do".
    - E.g.: deceleration rate when braking on wet road.
  - These learning tasks are more difficult if the environment is only partially observable.

Learning agents

- In previous learning forms, no need to access the external performance standard.
- The situation is more complex for a utility-based agent that wishes to **learn utility information**.
  - E.g. taxi to change its "driving style" based on the tip given as a **reward** (or not, as a **penalty**) by the clients.
  - The external performance standard must inform the agent that the loss of tips is bad for its overall performance.

  ➔ agent should be able to learn that violent maneuvers do not contribute to its own utility.
- ***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.*

How the components of agent programs work

- The environment representations can be placed along an axis of increasing complexity and expressive power—**atomic**, **factored**, and **structured.**
- **atomic representation:** each state of the world is indivisible—it has no internal structure.
  - Finding a driving route from one city to another via some sequence of cities.
  - It may suffice to reduce the state of world to just the name of the city we are in.
  - *Search* and *game-playing*, *Hidden Markov models* and *Markov decision processes* all work with atomic representations.

How the components of agent programs work

- For the same problem, if need for more than just atomic location in one city or another, such as how much gas is in the tank, current GPS coordinates, whether or not the oil warning light is working, amount of spare change available for toll crossings, and so on, then **Factored representation.**
- This representation splits up each state into a fixed set of **variables** / **attributes**, each of which can have a **value**.
- Two different atomic states have nothing in common while two different factored states can share some attributes.
  - Think of the difference between logic propositions and variables.
  - E.g. One GPS location or another; having lots of gas or having no gas; etc.

How the components of agent programs work

- Factored representation is finer and more expressive.
- *Uncertainty* can also be represented.
- Various areas of AI are based on factored representations, including constraint satisfaction algorithms, propositional logic, planning, and Bayesian networks.

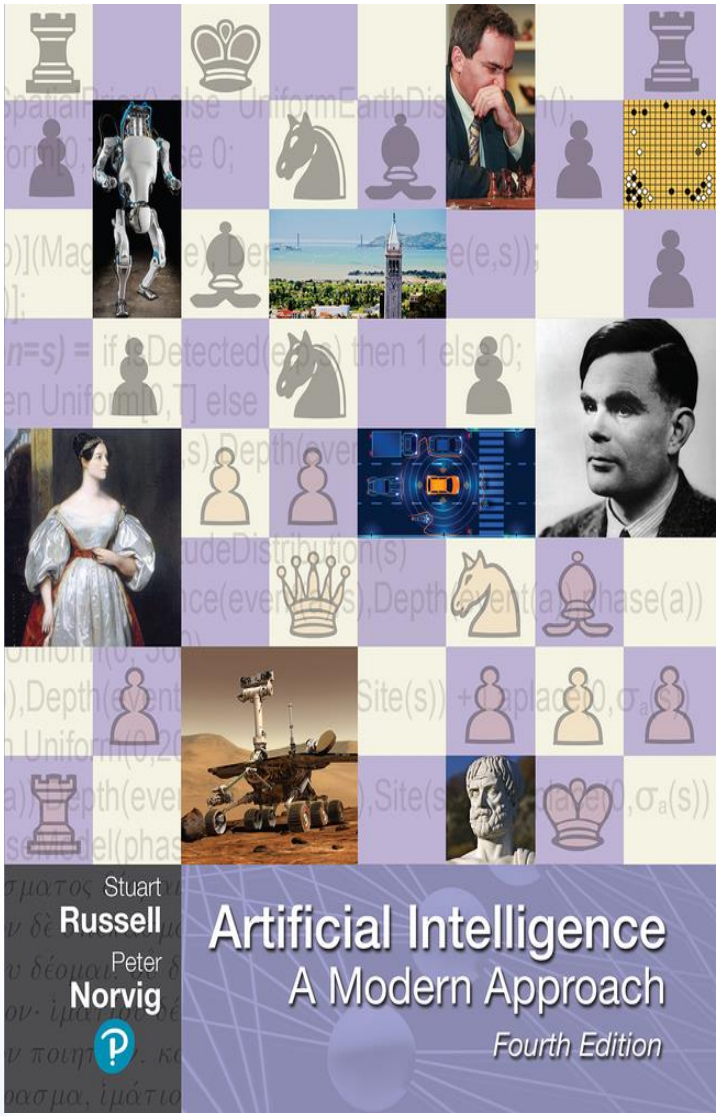
How the components of agent programs work

- 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.
- A factored representation cannot reasonably be pre-equipped with an attribute like *TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow* with value true or false.
- A **structured representation** is needed: objects such as cows and trucks and their various and varying relationships can be described explicitly.
- Structured representations underlie relational databases and first-order logic, first-order probability models, knowledge-based learning and much of natural language understanding.

Conclusion

- The axis along which atomic, factored, and structured representations lie is the axis of increasing **expressiveness.**
- As expressiveness increases more can be captured, and at least as concisely.
- Example: the rules of chess can be written in a page or two of a structured-representation language (e.g. FOL) but require thousands of pages when written in a factored-representation language (e.g. propositional logic).
- BUT, reasoning and learning become more complex as the expressive power of the representation increases.
- Sometimes several representations used at once.

Slides based on the textbook



- Russel, S. and Norvig, P. (2020) Artificial Intelligence, A Modern Approach (4th Edition), Pearson Education Limited.