In [1]:

```
1
```

In [10]:

```
1 from IPython.display import Video
```

Introduction to AI

Key Terminology ¶

- Intelligence : how we think and act
- AI : The field of artificial intelligence, or AI, is concerned with not just understanding but also building intelligent entities—machines that can compute how to act effectively and safely in a wide variety of novel situations.
  - fidelity to human performance (Acting Humanly)
  - rationality, doing the “right thing.”. Taking decisions that are always mathematically perfect.
  - a property of internal thought processes and reasoning,
  - intelligent behavior

Acting humanly: The Turing test approach

- The Turing test, proposed by Alan Turing (1950), was designed as a thought experiment that answers the question “Can a machine think?”
- A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.
- Programming a computer to pass a rigorously applied test provides plentyto work on. The computer would need the following capabilities:
  - natural language processing to communicate successfully in a human language;
  - knowledge representation to store what it knows or hears;
  - automated reasoning to answer questions and to draw new conclusions;

    - machine learning to adapt to new circumstances and to detect and extrapolate patterns.
- However, other researchers have proposed a total Turing test, which requires interaction with objects and people in the real world. To pass the total Turing test, a machine will need
    - computer vision and speech recognition to perceive the world;
    - robotics to manipulate objects and move about.
- These six disciplines compose most of AI.

Thinking humanly: The cognitive modeling approach

To say that a program thinks like a human, we must know how humans think. We can learn about human thought in three ways:

- introspection — trying to catch our own thoughts as they go by;
- psychological experiments — observing a person in action;
- brain imaging — observing the brain in action.

Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input-output behavior matches corresponding human behavior, that is evidence that some of the program's mechanisms could also be operating in humans.

Thinking rationally: The "laws of thought" approach

- The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking"— that is, irrefutable reasoning processes. His syllogisms provided patterns for argument structures that always yielded correct conclusions when given correct premises.
    - Socrates is a man
    - All men are mortal
    - Concludes that Socrates is mortal.
- These laws of thought were supposed to govern the operation of the mind; their study initiated the field called logic.

- Logicians in the 19th century developed a precise notation for statements about objects in the world and the relations among them. (Contrast this with ordinary arithmetic notation, which provides only for statements about numbers.)
- By 1965, programs could, in principle, solve any solvable problem described in logical notation.
- Logic as conventionally understood requires knowledge of the world that is certain — a condition that, in reality, is seldom achieved.
- We simply don't know the rules of, say, politics or warfare in the same way that we know the rules of chess or arithmetic.
- The theory of probability fills this gap, allowing rigorous reasoning with uncertain information.
- In principle, it allows the construction of a comprehensive model of rational thought, leading from raw perceptual information to an understanding of how the world works to predictions about the future.
- What it does not do, is generate intelligent behavior. For that, we need a theory of rational action. Rational thought, by itself, is not enough.

Acting rationally: The rational agent approach

- An agent is just something that acts (agent comes from the Latin agere, to do).
- Of course, all computer programs do something, but computer agents are expected to do more:
    - operate autonomously,
    - perceive their environment,
    - persist over a prolonged time period,
    - adapt to change, and
    - create and pursue goals.
- A rational agent is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.

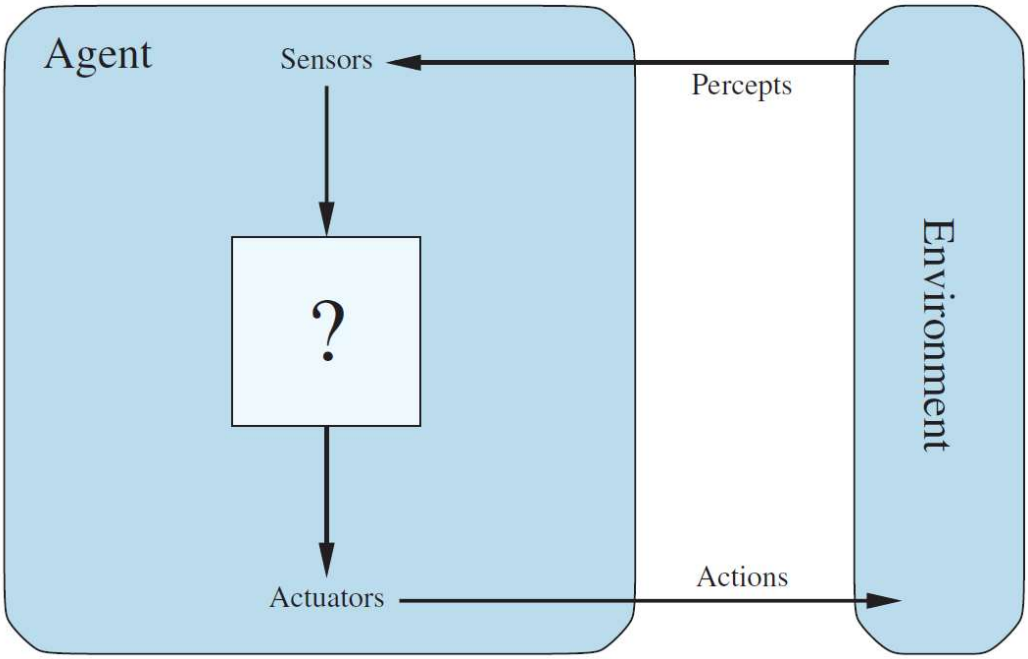
In the "laws of thought" approach to AI, the emphasis was on correct inferences. Making correct inferences is sometimes part of being a rational agent, because one way to act rationally is to deduce that a given action is best and then to act on that conclusion. On the

other hand, there are ways of acting rationally that cannot be said to involve inference. For example, recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.

All the skills needed for the Turing test also allow an agent to act rationally. Knowledge representation and reasoning enable agents to reach good decisions. We need to be able to generate comprehensible sentences in natural language to get by in a complex society. We need learning not only for erudition, but also because it improves our ability to generate effective behavior, especially in circumstances that are new.

Agents

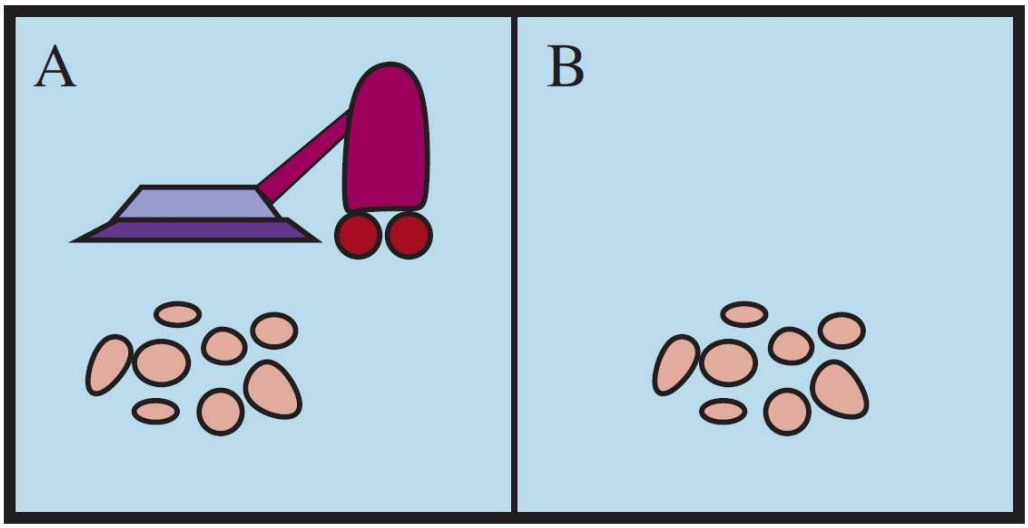
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
- A software agent receives file contents, network packets, and human input (keyboard/mouse/touchscreen/voice) as sensory inputs and acts on the environment by writing files, sending network packets, and displaying information or generating sounds.



- The environment is the part that affects what the agent perceives and that is affected by the agent's actions.
- Percept refers to the content an agent's sensors are perceiving.
- An agent's percept sequence is the complete history of everything the agent has ever perceived.
- An agent's choice of action at any given instant can depend on its builtin knowledge and on the entire percept sequence observed to date, but not on anything it hasn't perceived.
- An agent's behavior is described by the **agent function** that maps any given percept sequence to an action.
- The agent function for an artificial agent will be implemented by an agent program.
- The agent function is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

Example Agent

- A hand-held calculator is an agent that chooses the action of displaying "4" when given the percept sequence "2 + 2 =,"
- A vacuum-cleaner world with just two locations. Each location can be clean or dirty, and the agent can move left or right and can clean the square that it occupies.



- Partial tabulation of a simple agent function for the vacuum-cleaner world is shown in the following figure. Note that the table is of unbounded size unless there is a restriction on the length of possible percept sequences.

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

Rational Agent

- A rational agent is one that does the right thing.
- Consequentialism: we evaluate an agent's behavior by its consequences. When an agent is put in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well.
- 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.
- An **omniscient agent** knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.
- Rationality maximizes expected performance, while perfection maximizes actual performance.
- Doing actions in order to modify future percepts—sometimes called **information gathering**—is an important part of rationality.
    - An example of information gathering is provided by the exploration that must be undertaken by a vacuum-cleaning agent in an initially unknown environment.
- There are extreme cases in which the environment is completely known apriori and completely predictable. In such cases, the agent need not perceive or learn; it simply acts correctly.

Task Environment

- In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Specification of the performance measure, the environment, and the agent's actuators and sensors is called description of the **task environment. PEAS (Performance, Environment, Actuators, Sensors)**.
- The PEAS description for an automated taxi driver's task environment.
    - **Performance measure**: Desirable qualities include getting to the correct destination; minimizing fuel consumption and wear and tear; minimizing the trip time or cost; minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort; maximizing profits. Obviously, some of these goals conflict, so tradeoffs will be required.
    - **Environment**: Any taxi driver must deal with a variety of roads, traffic, pedestrians, stray animals, road works, police cars, puddles, and potholes. The taxi must also interact with potential and actual passengers. Obviously, the more restricted the environment, the easier the design problem.
    - **Actuators** for an automated taxi include those available to a human driver: control over the engine through the accelerator and control over steering and braking. In addition, it will need output to a display screen or voice synthesizer to talk back to the passengers, and perhaps some way to communicate with other vehicles, politely or otherwise.
    - **Sensors** for the taxi will include one or more video cameras so that it can see, as well as lidar and ultrasound sensors to detect distances to other cars and obstacles. To avoid speeding tickets, the taxi should have a speedometer, and to control the vehicle properly, especially on curves, it should have an accelerometer. To determine the mechanical state of the vehicle, it will need the usual array of engine, fuel, and electrical system sensors. Like many human drivers, it might want to access GPS signals so that it doesn't get lost. Finally, it will need touchscreen or voice input for the passenger to request a destination.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits, minimize impact on other road users	Roads, other traffic, police, pedestrians, customers, weather	Steering, accelerator, brake, signal, horn, display, speech	Cameras, radar, speedometer, GPS, engine sensors, accelerometer, microphones, touchscreen

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments	Touchscreen/voice entry of symptoms and findings
Satellite image analysis system	Correct categorization of objects, terrain	Orbiting satellite, downlink, weather	Display of scene categorization	High-resolution digital camera
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, tactile and joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, raw materials, operators	Valves, pumps, heaters, stirrers, displays	Temperature, pressure, flow, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, feedback, speech	Keyboard entry, voice

Properties of task environments

- **Fully observable environment**: If an agent's sensors give it access to the Fully observable complete state of the environment at each point in time, then we say that 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.
- 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. The agent's goals may still be achievable, sometimes with certainty.
- **Single-agent vs. multiagent**: An agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment.
  - **Which entities must be viewed as agents?**
    - 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 merely as an object behaving according to the laws of physics.
    - If B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior, then B can be cosidered as agent by A.
  - **Competitive multiagent environment**: For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure.
  - **Partially cooperative multiagent environment**: In the taxi-driving environment, avoiding collisions maximizes the performance measure of all agents, so it is a partially cooperative multiagent environment.
  - It is also partially competitive because, for example, only one car can occupy a parking space.
  - **Deterministic vs. nondeterministic**. If the next state of the environment is completely determined by the current state and the action executed by the agent(s), then we say the environment is deterministic; otherwise, it is nondeterministic.

- In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment.
- If the environment is partially observable, however, then it could appear to be nondeterministic.
- Taxi driving is clearly nondeterministic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires may blow out unexpectedly and one's engine may seize up without warning.
- **Episodic**: 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. Many classification tasks are episodic.
    - For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn't affect whether the next part is defective.
- **Sequential**: 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.
- Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.
- **Static vs. dynamic**: If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static.
- Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.
- Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet.
- If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **Semidynamic**.
    - Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next.
    - Chess, when played with a clock, is semidynamic.
    - Crossword puzzles are static.

- **Discrete vs. continuous**: The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
    - For example, the chess environment has a finite number of distinct states (excluding the clock). Chess also has a discrete set of percepts and actions.
    - Taxi driving is a continuous-state and continuous-time problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.
- **Known Known vs. 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, the outcomes (or outcome probabilities if the environment is nondeterministic) 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.
- The distinction between known and unknown environments is not the same as the one between fully and partially observable environments. It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over.
- Conversely, an unknown environment can be fully observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.
- The performance measure itself may be unknown, either because the designer is not sure how to write it down correctly or because the ultimate user—whose preferences matter—is not known.
    - For example, a taxi driver usually won't know whether a new passenger prefers a leisurely or speedy journey, a cautious or aggressive driving style.
    - A virtual personal assistant starts out knowing nothing about the personal preferences of its new owner. In such cases, the agent may learn more about the performance measure based on further interactions with the designer or user. This, in turn, suggests that the task environment is necessarily viewed as a multiagent environment.
- The hardest case is partially observable, multiagent, nondeterministic, sequential, dynamic, continuous, and unknown.

- Taxi driving is hard in all these senses, except that the driver's environment is mostly known. Driving a rented car in a new country with unfamiliar geography, different traffic laws, and nervous passengers is a lot more exciting.

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
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

The Structure of Agents

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

We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the agent architecture

agent = architecture + program

Agent programs

- The agent programs that we discuss in this course all have the same skeleton: they take the current percept as input from the sensors and return an action to the actuators.
- Notice the difference between the agent program, which takes the current percept as input, and the agent function, which may depend on the entire percept history. The agent program has no choice but to take 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.

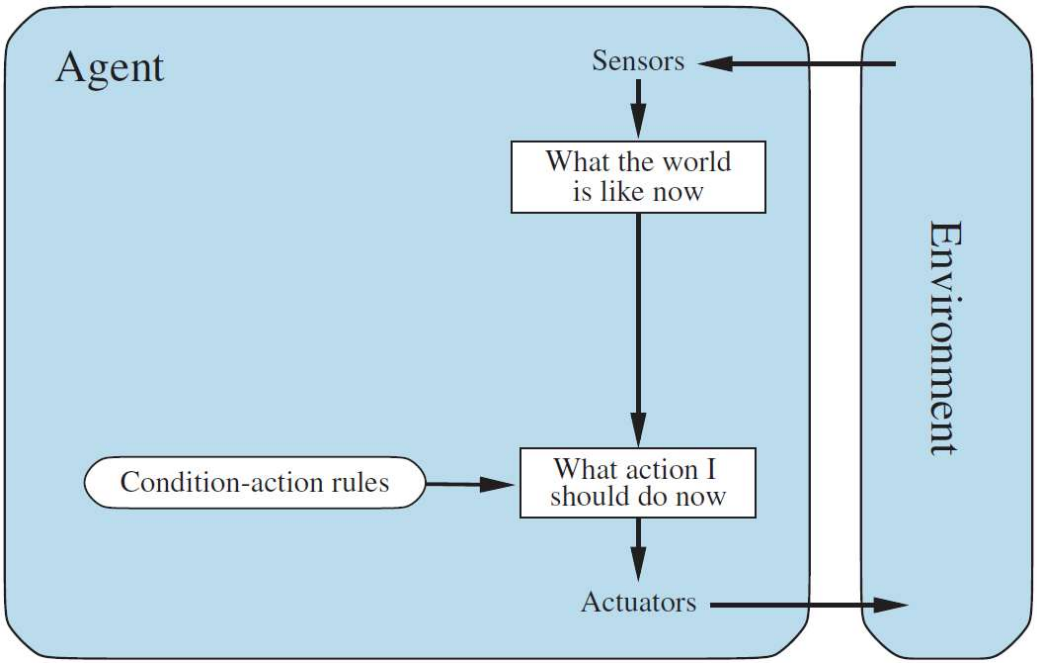
```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

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

- The table-driven approach to agent construction is doomed to failure. Let P be the set of possible percepts and let T be the lifetime of the agent (the total number of percepts it will receive).
- Consider the automated taxi: the visual input from a single camera (eight cameras is typical) comes in at the rate of roughly 70 megabytes per second (30 frames per second, 1080 X 720 pixels with 24 bits of color information). This gives a lookup table with over $10^{600;000;000;000}$ entries for an hour's driving. Even the lookup table for chess —a tiny, well-behaved fragment of the real world—has (it turns out) at least 10^{150} entries.
- The daunting size of these tables means that (a) no physical agent in this universe will have the space to store the table; (b) the designer would not have time to create the table; and (c) no agent could ever learn all the right table entries from its experience.

Types of agent programs

Simple reflex agents

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




```
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
```

```
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 mapping between percept and action is called condition–action rule,

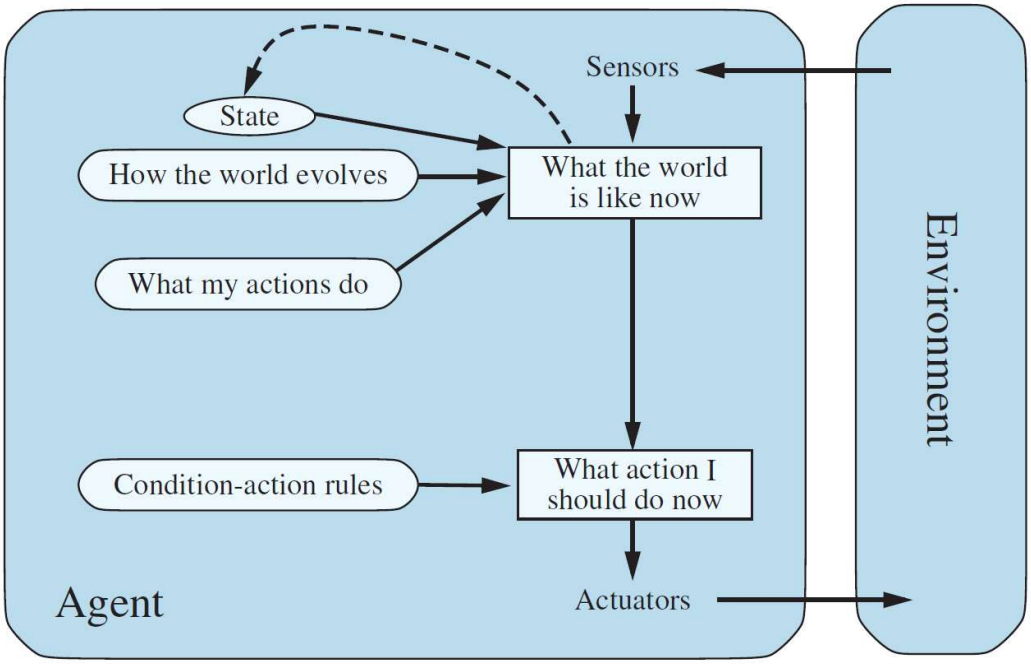
  if car-in-front-is-braking then initiate-braking

- Simple reflex agents will work only if the correct decision can be made on the basis of just the current percept—that is, only if the environment is fully observable.
- Even a little bit of unobservability can cause serious trouble.
  - For example, the braking rule given earlier assumes that the condition car-in-front-is-braking can be determined from the current percept—a single frame of video. This works if the car in front has a centrally mounted (and hence uniquely identifiable) brake light. Unfortunately, older models have different configurations

of taillights, brake lights, and turn-signal lights, and it is not always possible to tell from a single image whether the car is braking or simply has its taillights on. A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all.

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 Internal state aspects of the current state.



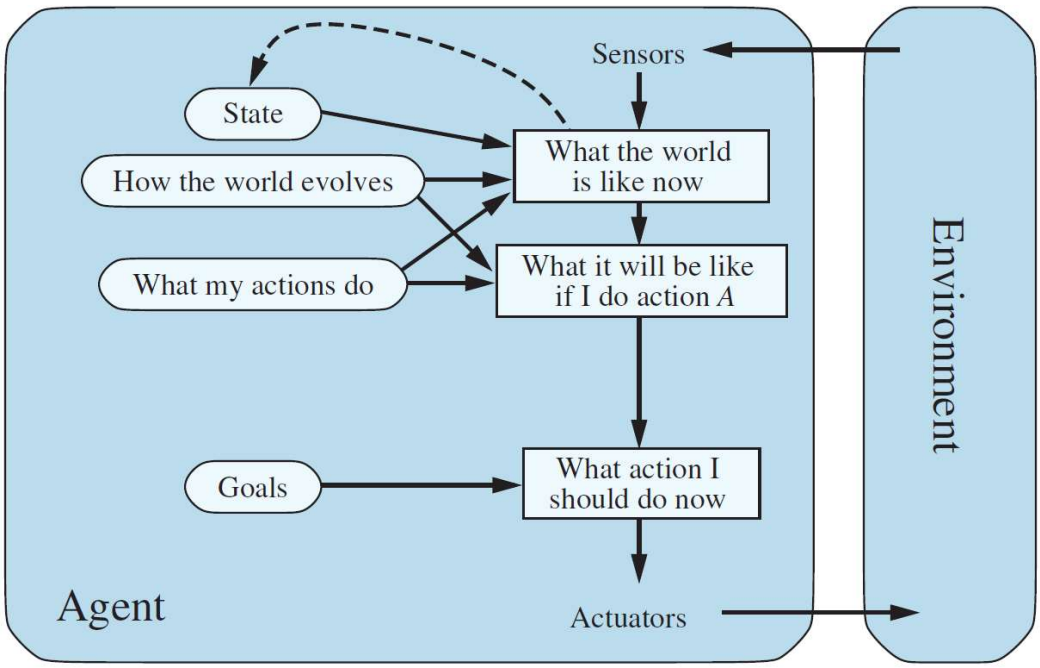
- First, we need some information about how the world changes over time, which can be divided roughly into two parts: the effects of the agent's actions and how the world evolves independently of the agent.
- This knowledge about "how the world works" is called a **transition model of the world**.
- Second, we need some information about how the state of the world is reflected in the agent's percepts. This kind of knowledge is called a **sensor model**.
- Together, the transition model and sensor model allow an agent to keep track of the state of the world—to the extent possible given the limitations of the agent's sensors.
- An agent that uses such models is called a model-based agent.

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
  persistent: *state*, the agent's current conception of the world state
    transition_model, a description of how the next state depends on the current state and action
    sensor_model, a description of how the current world state is reflected in the agent's percepts
    rules, a set of condition–action rules
    action, the most recent action, initially none

  state ← UPDATE-STATE(*state*, *action*, *percept*, *transition_model*, *sensor_model*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← *rule*.ACTION
  return *action*

Goal-based agents

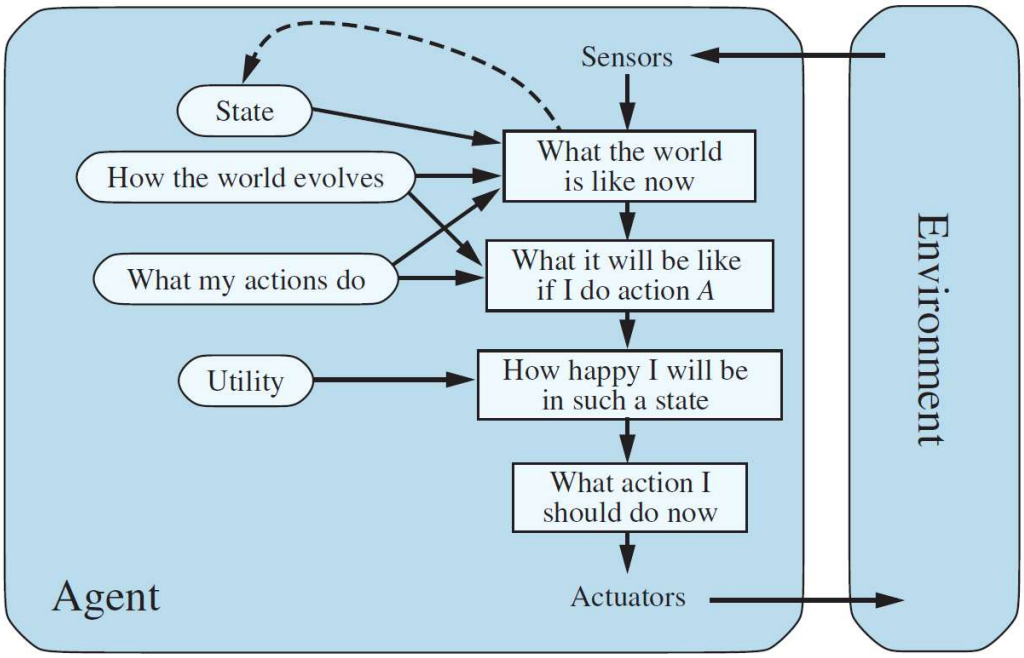
- Knowing something about the current state of the environment is not always enough to decide what to do.
  - For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.
- The agent program can combine this with the model (the same information as was used in the model-based reflex agent) to choose actions that achieve the goal.



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

Utility-based agents

- Goals alone are not enough to generate high-quality behavior in most environments.
  - For example, many action sequences will get the taxi to its destination (thereby achieving the goal), but some are quicker, safer, more reliable, or cheaper than others.
- An agent's utility function is essentially an internalization of the performance measure. Provided that the internal utility function and the external performance measure are in agreement, an agent that chooses actions to maximize its utility will be rational according to the external performance measure.
- Like goal-based agents, a 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.
  - 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.



- Partial observability and nondeterminism are ubiquitous in the real world, and so, therefore, is decision making under uncertainty. Technically speaking, a rational utility-

based agent chooses the action that maximizes the expected utility of the action outcomes—that is, the utility the agent expects to derive, on average, given the probabilities and utilities of each outcome.

In [9]:

```
1 Video(data = './videos/Micromouse_Maze.webm', width = 600)
```

Out[9]: