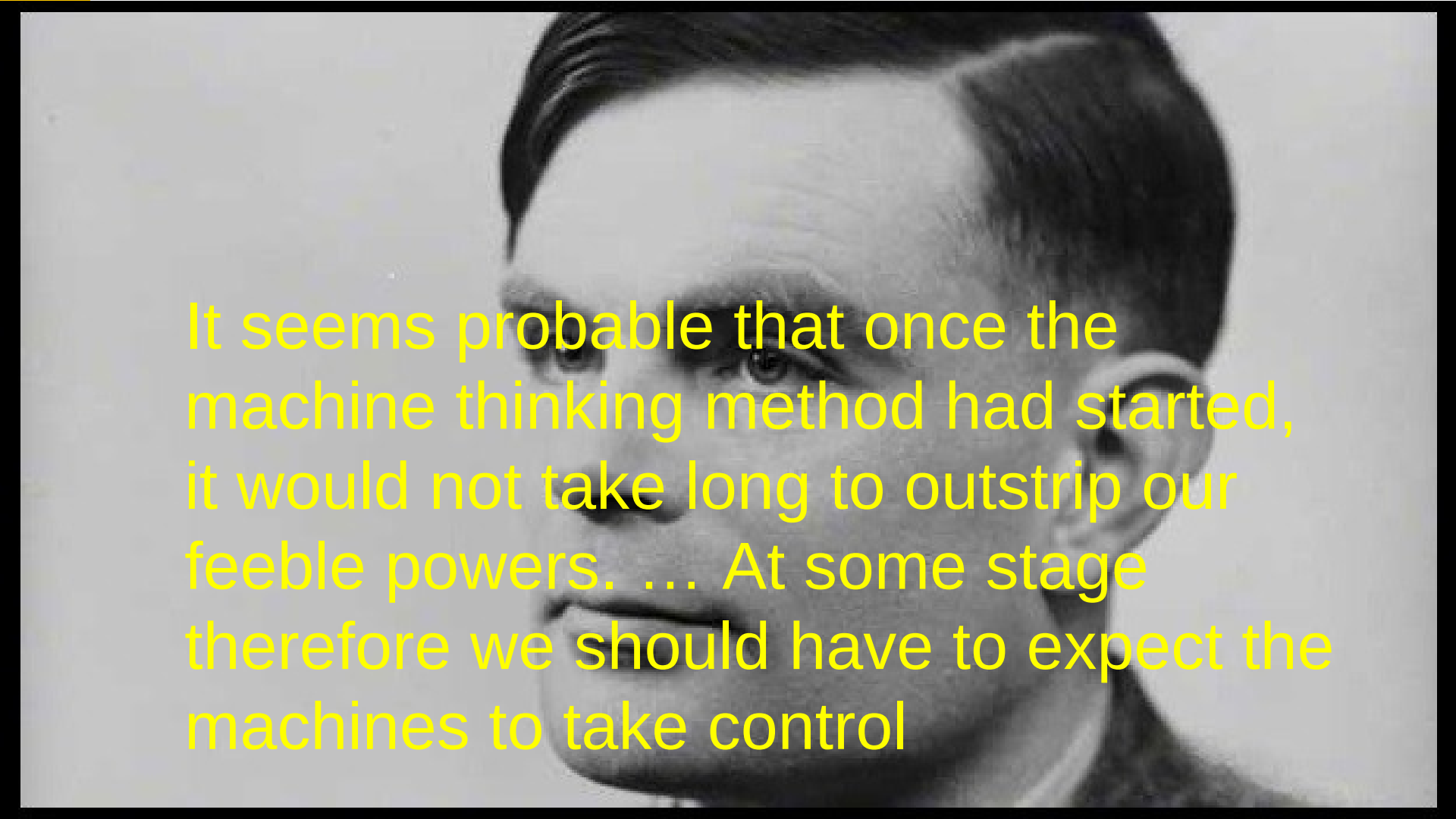


Announcements

- Upcoming due dates
 - Preliminary Survey (on Canvas) due Wednesday 26 Jan 10:59 pm EST

Future

- We are doing AI...
 - To create intelligent systems
 - The more intelligent, the better
 - To gain a better understanding of human intelligence
 - To magnify those benefits that flow from it
 - E.g., net present value of human-level AI \geq \$13,500T
 - Might help us avoid war and ecological catastrophes, achieve immortality and expand throughout the universe
- What if we succeed?



It seems probable that once the machine thinking method had started, it would not take long to outstrip our feeble powers. ... At some stage therefore we should have to expect the machines to take control

What's bad about better AI?

- AI that is incredibly good at achieving something other than what we really want
- AI, economics, statistics, operations research, control theory all assume utility to be *fixed, known, and exogenously specified*
 - Machines are intelligent to the extent that their actions can be expected to achieve their objectives
 - Machines are beneficial to the extent that their actions can be expected to achieve our objectives

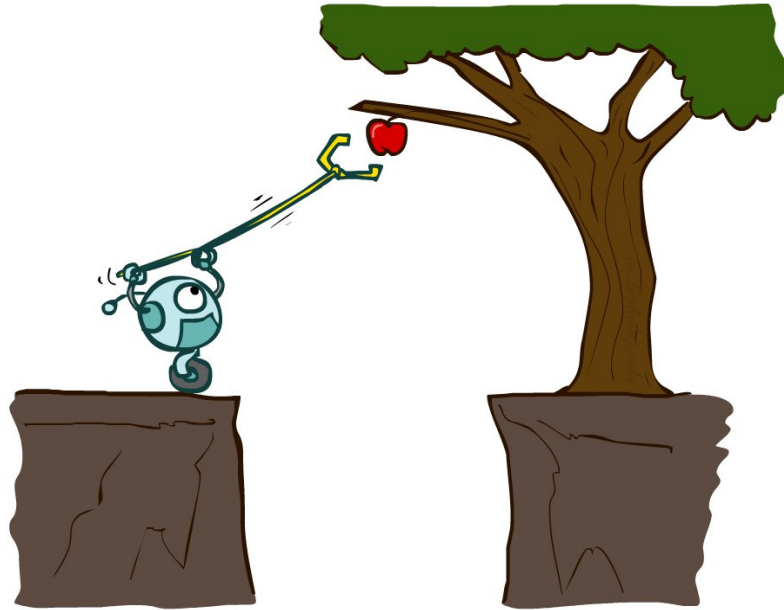
A new model for AI

1. The machine's only objective is to maximize the realization of human preferences
2. The robot is initially uncertain about what those preferences are
3. Human behavior provides evidence about human preferences

“The essential task of our age” [Nick Bostrom, Professor of Philosophy, Oxford]

CS 370: Artificial Intelligence

Agents and environments



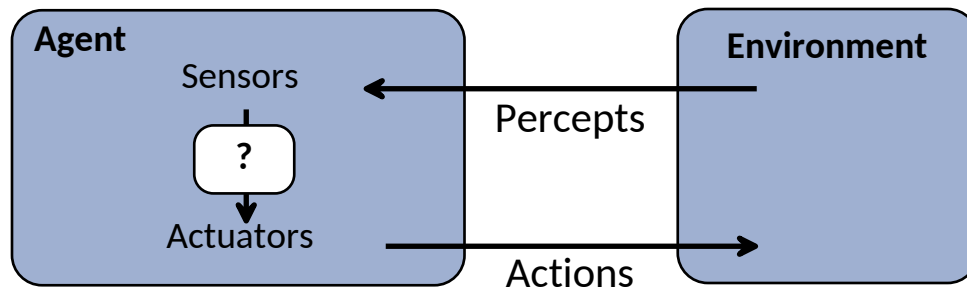
Instructor: Dr. William DeMeo

Slides courtesy of Stuart Russell and Dawn Song, ai.berkeley.edu

Outline

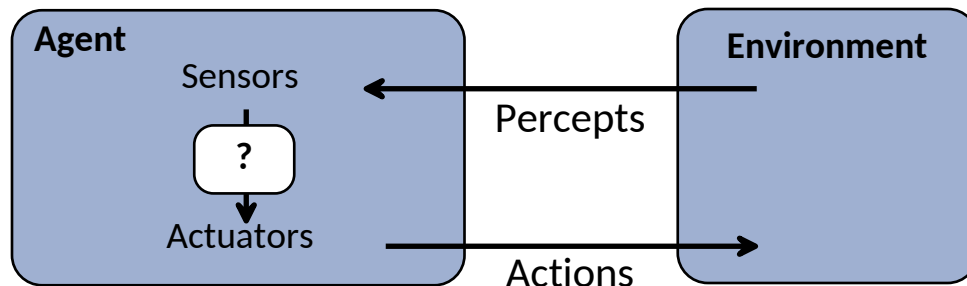
- Agents and environments
- Rationality
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Environment types
- Agent types

Agents and environments



- An agent *perceives* its environment through *sensors* and *acts* upon it through *actuators* (or *effectors*, depending on whom you ask)

Agents and environments



- Are humans agents?
- Yes!
 - Sensors = vision, audio, touch, smell, taste, proprioception
 - Actuators = muscles, secretions, changing brain state

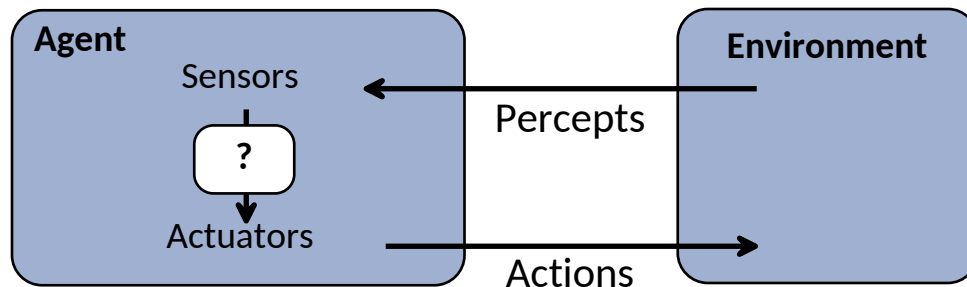
Proprioception

Proprioception, or *kinesthesia*

body's ability to sense movement, action, and location

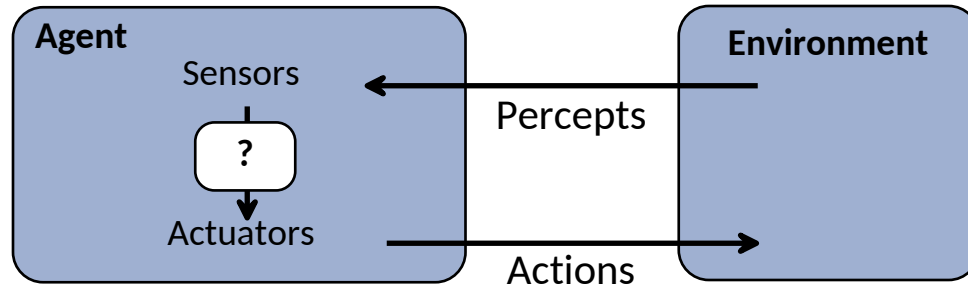
- present in every muscle movement
- allows walking without consciously thinking where next to place your foot
- lets you touch your elbow with your eyes closed

Agents and environments



- Are pocket calculators agents?
- Yes!
 - Sensors = key state sensors
 - Actuators = digit display

Agents and environments

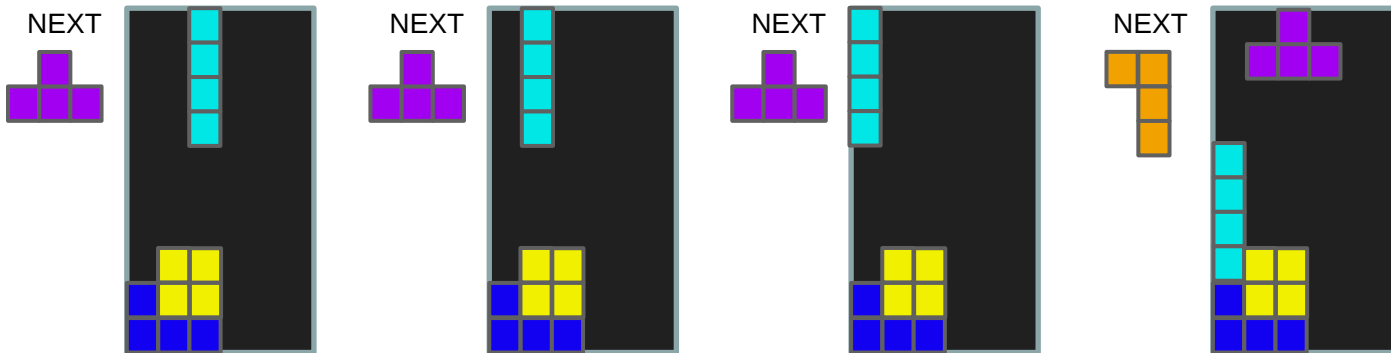


- AI is more interested in agents with large computational resources and environments that require nontrivial decision making

Agent functions

- The **agent function** maps from percept histories to actions:
 - $f: \mathcal{P}^* \rightarrow A$
 - I.e., the agent's actual response to any sequence of percepts

Percept



Action

LEFT

LEFT

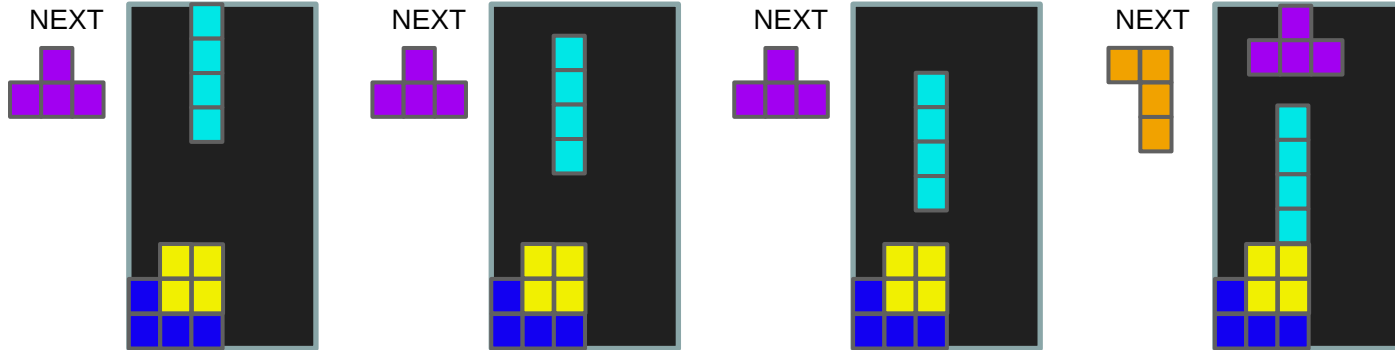
DROP

RIGHT

Agent programs

- The **agent program** I runs on some machine M to implement f :
 - $f = \text{Agent}(I, M)$
 - Real machines have limited speed and memory, introducing delay, so agent function f depends on M as well as I

Percept



Action

NOOP

NOOP

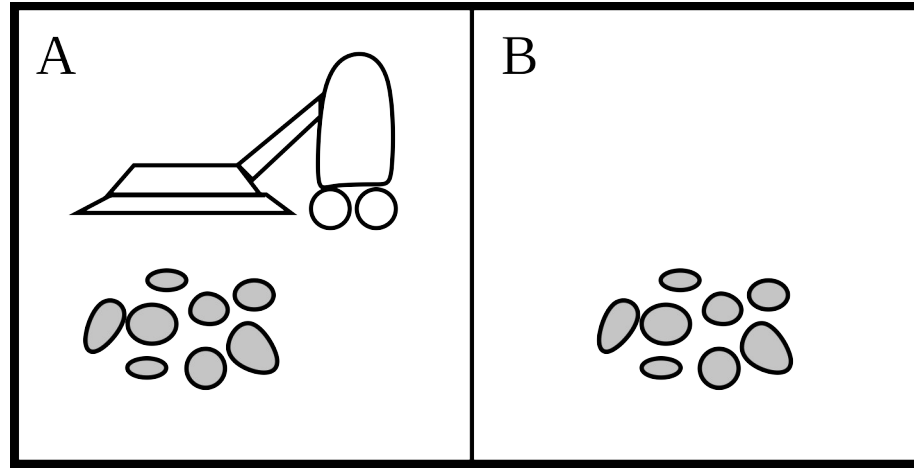
NOOP

LEFT

Agent functions and agent programs

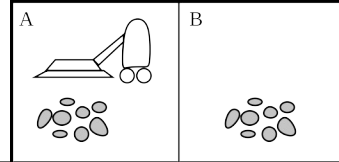
- Can every agent function be implemented by some agent program?
 - No! Consider agent for halting problems, NP-hard problems, chess with a slow PC

Example: Vacuum world



- Percepts: [location,status], e.g., [A,Dirty]
- Actions: *Left, Right, Suck, NoOp*

Vacuum cleaner agent



Agent function

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

Agent program

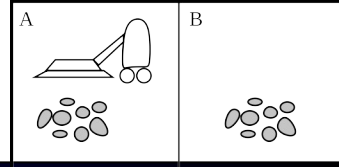
```
function Reflex-Vacuum-Agent([location,status])  
    returns an action  
if status = Dirty then return Suck  
else if location = A then return Right  
else if location = B then return Left
```

What is the *right* agent function?

Can it be implemented by a small agent program?

(Can we ask, “What is the right agent program?”)

Rationality



- Fixed **performance measure** evaluates the environment sequence
 - one point per square cleaned up?
 - NO! Rewards an agent who dumps dirt and cleans it up
 - one point per clean square per time step, for $t = 1, \dots, T$
- A **rational agent** chooses whichever action maximizes the expected value of the performance measure,
 - given the percept sequence to date and prior knowledge of environment.

Rational Agents

Does the Reflex-Vacuum-Agent implement a rational agent function?

Yes, if movement is free, or new dirt arrives frequently.

Rationality, contd.

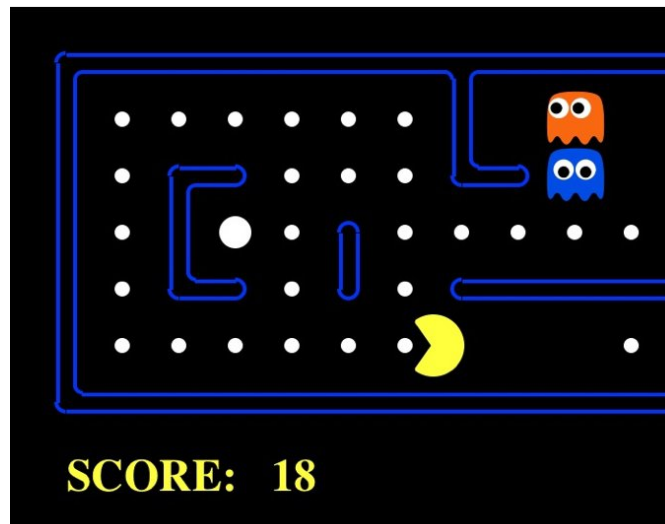
- Are rational agents **omniscient**?
 - No – they are limited by the available percepts
- Are rational agents **clairvoyant**?
 - No – they may lack knowledge of the environment dynamics
- Do rational agents **explore** and **learn**?
 - Yes – in unknown environments these are essential
- Do rational agents **make mistakes**?
 - No – but their actions may be unsuccessful
- Are rational agents **autonomous** (i.e., transcend initial program)?
 - Yes – as they learn, their behavior depends more on their own experience

A human agent in Pacman



The task environment - PEAS

- Performance measure
 - -1 per step; + 10 food; +500 win; -500 die; +200 hit scared ghost
- Environment
 - Pacman dynamics (incl ghost behavior)
- Actuators
 - Left Right Up Down
- Sensors
 - Entire state is visible (except power pellet duration)



PEAS: Automated taxi

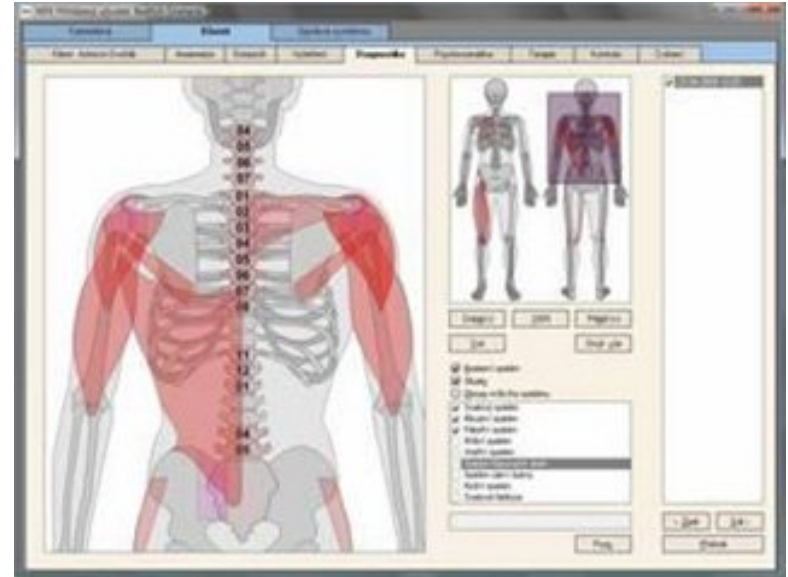
- Performance measure
 - Income, happy customer, vehicle costs, fines, insurance premiums
- Environment
 - US streets, other drivers, customers, weather, police...
- Actuators
 - Steering, brake, gas, display/speaker
- Sensors
 - Camera, radar, accelerometer, engine sensors, microphone, GPS



Image: <http://nypost.com/2014/06/21/how-google-might-put-taxi-drivers-out-of-business/>

PEAS: Medical diagnosis system

- Performance measure
 - Patient health, cost, reputation
- Environment
 - Patients, medical staff, insurers, courts
- Actuators
 - Screen display, email
- Sensors
 - Keyboard/mouse



Environment types

Fully Observable vs. Partially Observable

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is ***fully observable***.

A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action. (Relevance depends on the performance measure).

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.

Environment types

Fully Observable vs Partially Observable

Examples of *partially observable* environments.

1. vacuum agent with only a local dirt sensor (can't tell if there is dirt in other squares)
2. automated taxi (can't see what other drivers are thinking)

If the agent has no sensors at all then the environment is *unobservable*.

One might think that in such cases the agent's plight is hopeless, but we'll see (in Ch 4) that agent's goals may still be achievable, sometimes with certainty.

Environment types

SINGLE-AGENT VS. MULTIAGENT

The distinction between *single-agent* and *multiagent* environments seems self-evident.

For example,

- 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.

However, there are some subtle issues.

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?

When is an entity an agent?

The key question:

Is B 's behavior best described as maximizing a performance measure (whose value depends on agent A's behavior)?

If the answer is yes, then B is an agent.

Environment types

Deterministic vs. Stochastic

If the next state of the environment is completely determined by the current state and the action executed by the agent(s), then we call the environment ***deterministic***.

Otherwise, it is a ***nondeterministic*** (or stochastic) environment.

Environment types

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 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.

Environment types

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 are hard; they continuous ask the agent what it wants to do; if it hasn't decided, that counts as deciding to do nothing.

If the environment does not change with the passage of time, but the agent's performance score does, then we say the environment is **semidynamic**.

Environment types

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.

EXAMPLES

- 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, but typically treated as representing continuously varying intensities and locations.

Environment types

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.

Environment types

the easiest and hardest cases

EASIEST

fully observable, single-agent, deterministic, episodic, static, discrete, and known

HARDEST

partially observable, multiagent, nondeterministic, sequential, dynamic, continuous, unknown

Taxi driving environment is hardest, except driver's environment is often mostly known.

Driving a rented car in a new country with unfamiliar geography, different traffic laws, and nervous passengers is a lot more exciting!

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
English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

Environment types

	Pacman	Backgammon	Diagnosis	Taxi
Fully or partially observable				
Single-agent or multiagent				
Deterministic or stochastic				
Static or dynamic				
Discrete or continuous				
Known physics?				
Known perf. measure?				

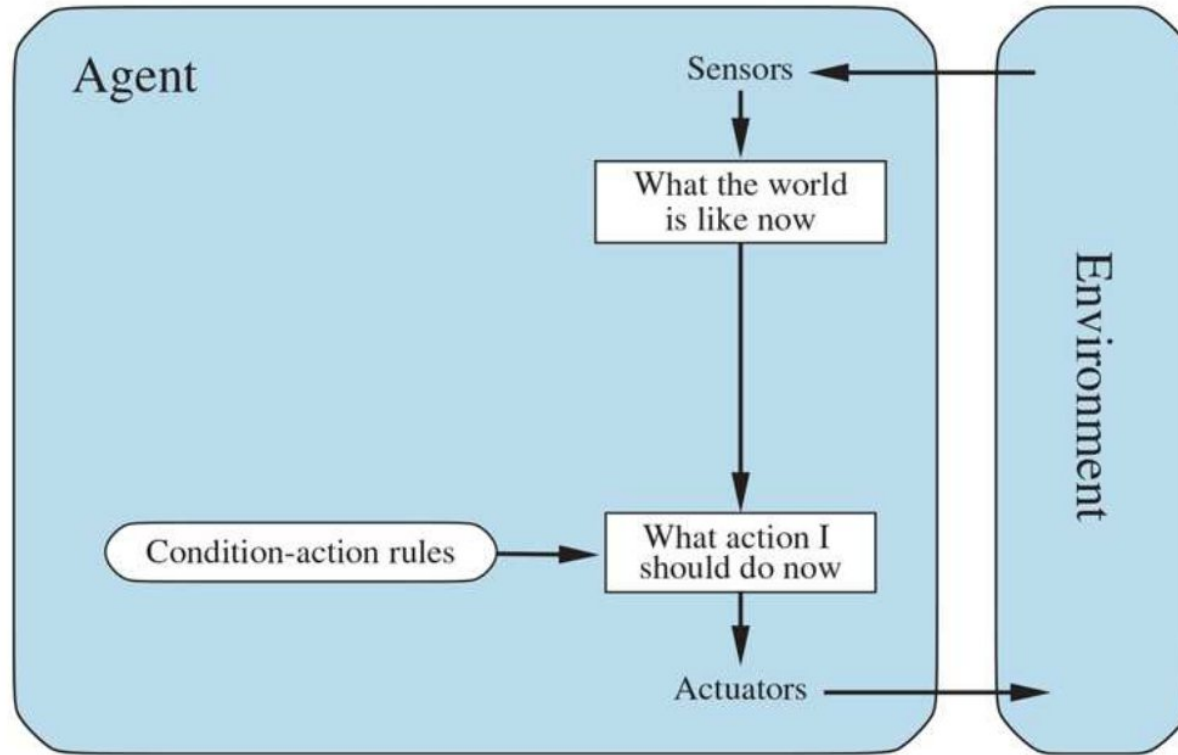
Agent design

- The environment type largely determines the agent design
 - *Partially observable* => agent requires *memory* (internal state)
 - *Stochastic* => agent may have to prepare for *contingencies*
 - *Multi-agent* => agent may need to behave *randomly*
 - *Static* => agent has time to compute a rational decision
 - *Continuous time* => continuously operating *controller*
 - *Unknown physics* => need for *exploration*
 - *Unknown perf. measure* => observe/interact with *human principal*

Agent types

- In order of increasing generality and complexity
 - Simple reflex agents
 - Reflex agents with state
 - Goal-based agents
 - Utility-based agents

Simple reflex agents



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

Pacman agent in Python

```
class GoWestAgent(Agent):
```

```
    def getAction(self, percept):
```

```
        if Directions.WEST in percept.getLegalPacmanActions():
```

```
            return Directions.WEST
```

```
        else:
```

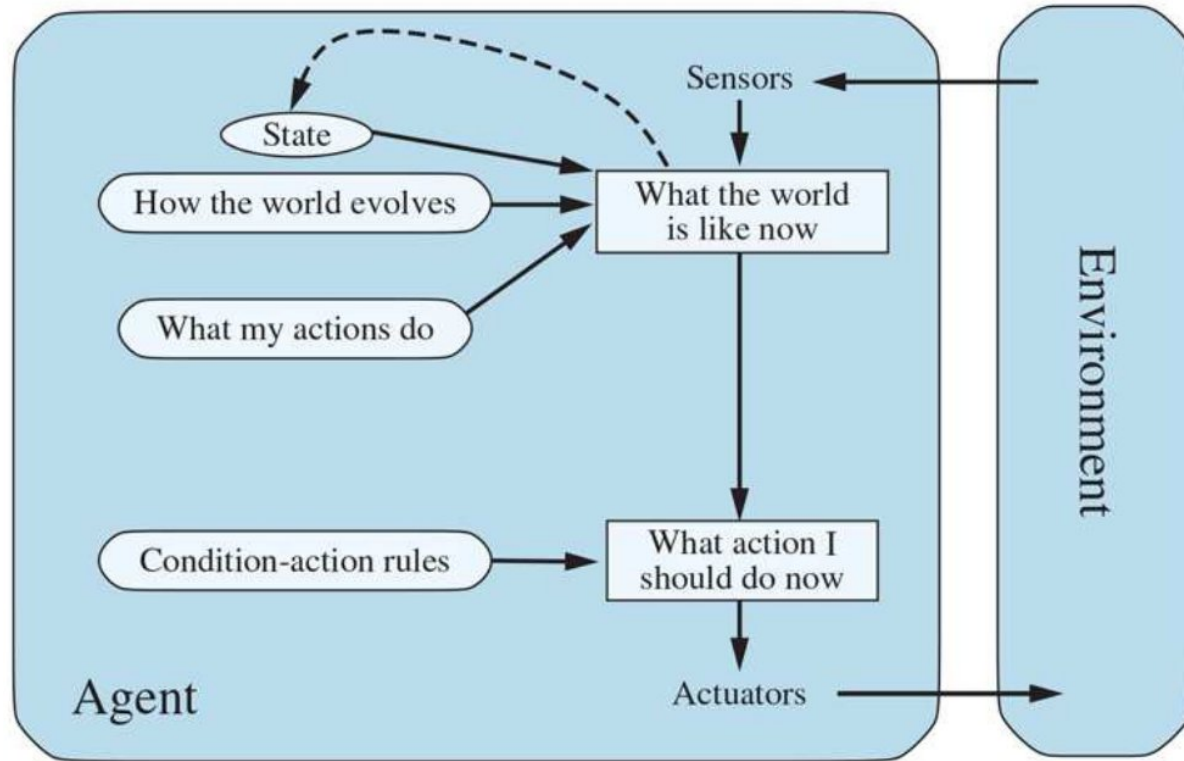
```
            return Directions.STOP
```


Pacman agent contd.

- Can we (in principle) extend this reflex agent to behave well in all standard Pacman environments?
 - No – Pacman is not quite fully observable (power pellet duration)
 - Otherwise, yes – we can (in principle) make a lookup table.....

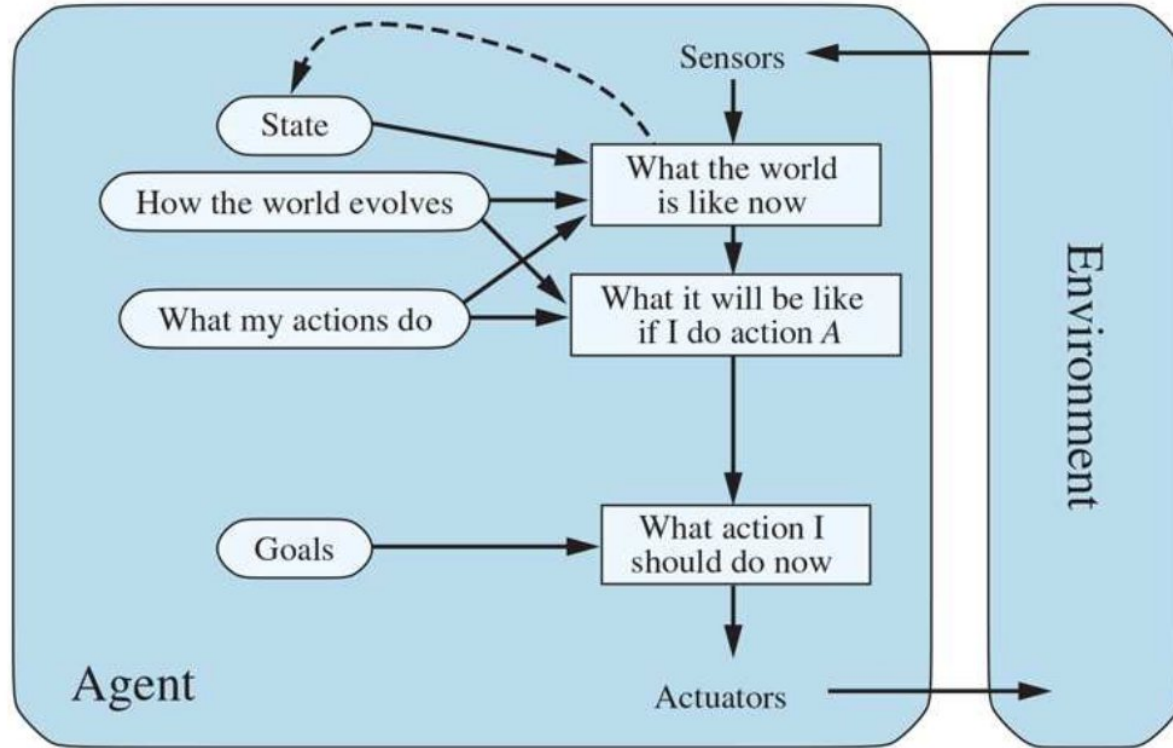
Model-based Reflex Agent

with state



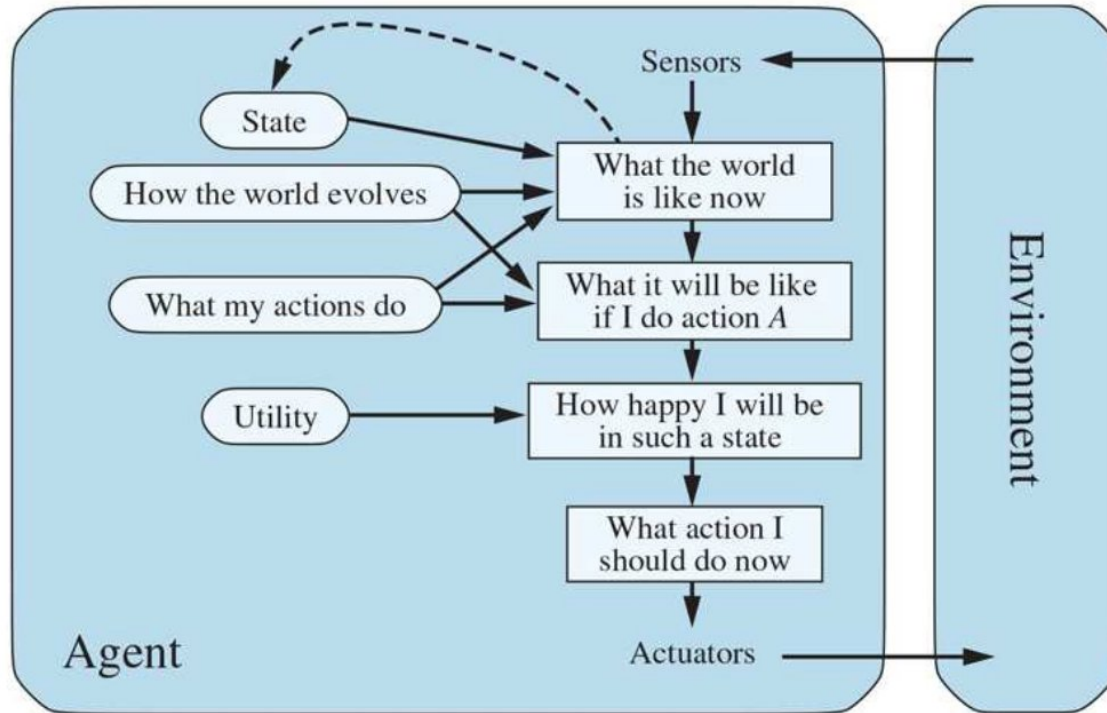
A model-based reflex agent.

Model-based, Goal-based Agent



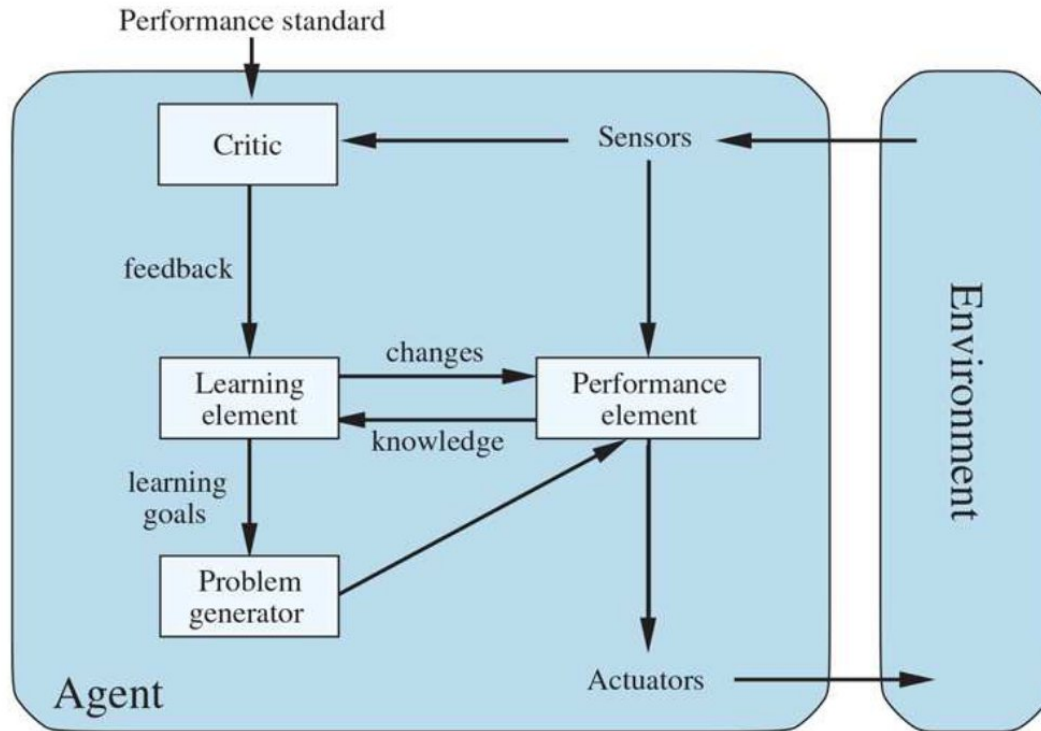
A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

Utility-based agents



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 Agent



A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

How components of agent programs work

We described agent programs (in high-level terms) as consisting of various components which aim to answer questions such as:

- “What is the world like now?”
- “What action should I do now?”
- “What do my actions do?”

The next question is, “How on Earth do these components work?”

It takes about a thousand pages to begin to answer that question properly, but we can draw some crude distinctions between ways in which components can represent the task environment: (in order of increasing complexity and expressive power)

1. atomic
2. factored
3. structured.

Atomic representation of the task environment

In an *atomic representation* each state of the world is indivisible—it has no internal structure.

Consider the task 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 the 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.

The standard algorithms underlying *search* and *game-playing* (Chs 3–5), *hidden Markov models* (Ch 14), and *Markov decision processes* (Ch 17) all work with atomic reps.

Factored representation of the task environment

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 driving problem, where we're concerned with more than just atomic location in a city.

We might need to pay attention to amount of gas in the tank, GPS coordinates, whether or not oil warning light works, money for tolls, what's playing on the radio, etc.

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); this makes it much easier to work out how to turn one state into another.

Factored representation of the task environment

Many important areas of AI are based on factored representations, including constraint satisfaction algorithms (Ch 6), propositional logic (Ch 7), planning (Ch 11), Bayesian networks (Chs 12–16), and various machine learning algorithms.

Structured representation of the task environment

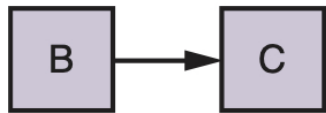
For many purposes, we need to understand the world as having things in it that are related to each other, not just variables with values.

Example: a large truck is ahead of us and reversing into driveway of a dairy farm; a loose cow blocks the truck's path. A factored representation is unlikely to come equipped with true/false attribute `TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow`.

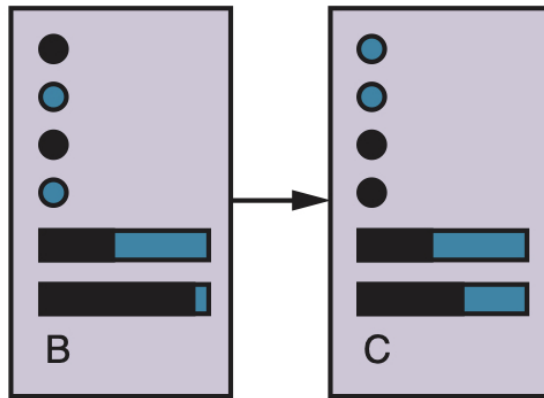
structured representation: objects (such as cows and trucks) and their various and varying relationships can be described explicitly.

These underlie relational databases and first-order logic (Chs 8—10), first-order probability models (Ch 15), and much of natural language understanding (Chs 23—24). Much of what humans express in nat language concerns objects and their relationships.

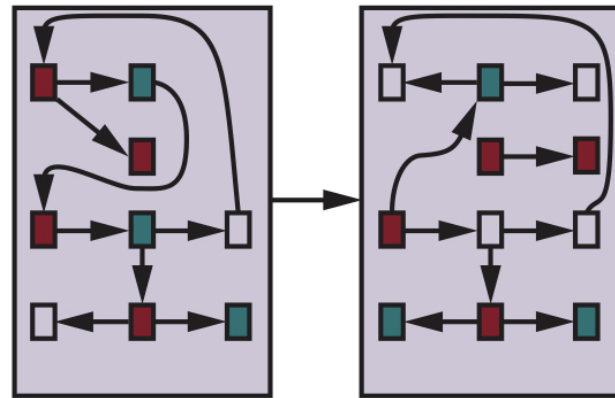
Spectrum of representations



(a) Atomic



(b) Factored



(c) Structured

Summary

- An **agent** interacts with an **environment** through **sensors** and **actuators**
- The **agent function**, implemented by an **agent program** running on a **machine**, describes what the agent does in all circumstances
- Rational agents choose actions that maximize their expected utility
- PEAS descriptions define task environments; precise PEAS specifications are essential and strongly influence agent designs
- More difficult environments require more complex agent designs and more sophisticated representations