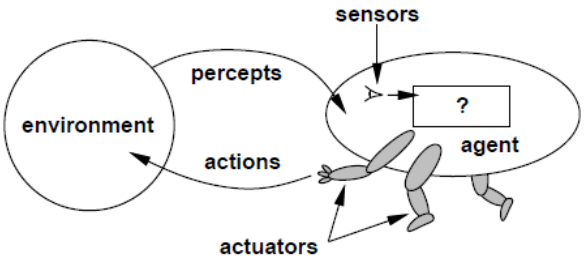
Intelligent Agents

Chapter 2

Agents

- An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**
- Human agent: eyes, ears, and other organs for sensors; hands, legs, mouth, and other body parts for actuators
- Robotic agent: cameras and infrared range finders for sensors; various motors for actuators
- Software agent? E.g. spell checker

Agents and Environments



Agents include humans, robots, softbots, thermostats, etc.

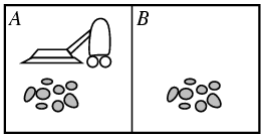
The agent function maps from percept histories to actions:

$f : \mathcal{P}^* \rightarrow \mathcal{A}$

The agent program runs on the physical architecture to produce f

Agent = architecture + program

Vacuum Cleaner World



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

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

What is the right way to fill out this table?

Can it be realistically implemented?

Rationality

- An agent should strive to "do the right thing", based on what it can perceive and the actions it can perform. The right action is the one that will cause the agent to be most successful
- Performance measure: An objective criterion for success of an agent's behavior
- E.g., performance measure of a vacuum-cleaner agent could be amount of dirt cleaned up, amount of time taken, amount of electricity consumed, amount of noise generated, etc.

Rational Agent

- 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.

Rational Agents

Fixed performance measure evaluates the environment sequence
- one point per square cleaned up in time T?
- one point per clean square per time step, minus one per move?
- penalize for $> k$ dirty squares?

A rational agent chooses whichever action maximizes the expected value of the performance measure given the percept sequence to date

Rational $\neq$ omniscient
- percepts may not supply all relevant information

Rational $\neq$ clairvoyant
- action outcomes may not be as expected

Hence, rational $\neq$ successful

Rational $\Rightarrow$ exploration, learning, autonomy

PEAS

- PEAS: Performance measure, Environment, Actuators, Sensors
- Must first specify the setting for intelligent agent design

Consider, e.g., the task of designing an automated taxi driver
- Performance measure
  - Safe, fast, legal, comfortable trip, maximize profits
- Environment
  - Roads, other traffic, pedestrians, customers
- Actuators
  - Steering wheel, accelerator, brake, signal, horn
- Sensors
  - Cameras, sonar, speedometer, GPS, odometer, engine sensors, accelerometer

PEAS

- Agent: Medical diagnosis system
  - Performance measure: Healthy patient, minimize costs, lawsuits
  - Environment: Patient, hospital, staff
  - Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)
  - Sensors: Keyboard (entry of symptoms, findings, patient's answers)

PEAS

- Agent: Part-picking robot
  - Performance measure: Percentage of parts in correct bins
  - Environment: Conveyor belt with parts, bins
  - Actuators: Jointed arm and hand
  - Sensors: Camera, joint angle sensors

PEAS

- Agent: Interactive English tutor
  - Performance measure: Maximize student's score on test
  - Environment: Set of students
  - Actuators: Screen display (exercises, suggestions, corrections)
  - Sensors: Keyboard (student answers)

PEAS

- Agent: Internet Shopping Agent
  - Performance measure: ?
  - Environment: ?
  - Actuators: ?
  - Sensors: ?

Environment types

- Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)
- Episodic (vs. sequential): The agent's experience is divided into atomic "episodes" (each episode consists of the agent perceiving and then performing a single action), and the choice of action in each episode depends only on the episode itself.

Environment types

- Static (vs. dynamic): The environment is unchanged while an agent is deliberating. (The environment is semidynamic if the environment itself does not change with the passage of time but the agent's performance score does)
- Discrete (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- Single agent (vs. multiagent): An agent operating by itself in an environment. Other “objects” should be agents if their behavior is maximized depending on the single agent’s behavior

Environments

Task Environment	Obser-vable	Deter-ministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess (no clock)	Fully*	Strategic	Sequential	Static	Discrete	Multi
Draw Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Taxi Driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Categorize Satellite Image	Fully	Deterministic	Episodic	Static/Semi	Continuous	Single
Internet Shopping Agent						
Real World						

* Not quite fully observable; why not?

The environment type largely determines the agent design

Agent functions and programs

- An agent is completely specified by the agent function mapping percept sequences to actions
- One agent function (or a small equivalence class) is rational
- Aim: find a way to implement the rational agent function concisely

Table-Driven Agent

```
function Table-Driven-Agent(percept) returns an action
    static:  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. Why?

Table-Driven Agent

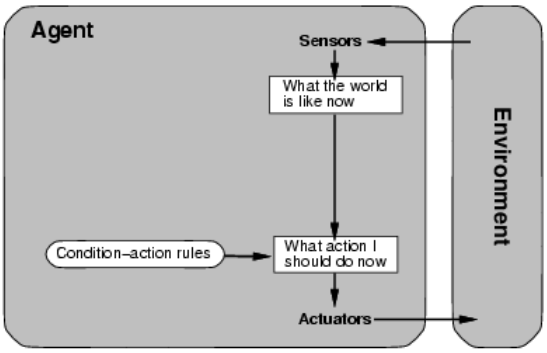
- If it were feasible, the table-driven agent does do what we want it to do
- Challenge of AI
  - Find out how to write programs that produce rational behavior from a small amount of code rather than a large number of table entries
  - Schoolchildren used to look up tables of square roots, but now a 5 line program for Newton's method is implemented on calculators

Agent types

- Four basic types in order of increasing generality:

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

Simple reflex agents



Selects actions on the basis of the current precept, ignoring the rest of the percept history.

Vacuum World Reflex Agent

```
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
```

Much smaller than the table – from ignoring percept history
In general, we match **condition-action rules** (if-then rules).

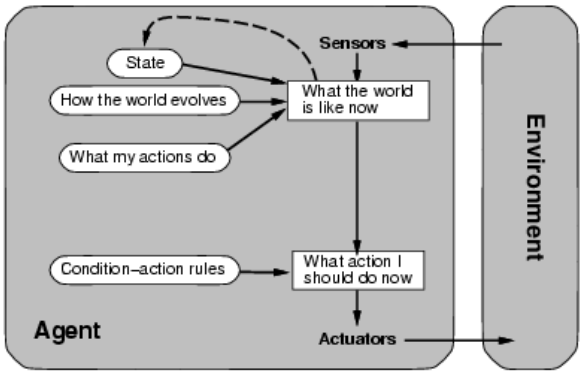
function Simple-Reflex-Agent(*percept*) **returns** an action
static: *rules*, a set of condition-action rules

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

Simple Reflex Agents

- Simple, but limited intelligence
- Only works well if the correct decision can be made on the basis of only the current percept
  - OK if environment fully observable
- A little partial observability can doom these agents
  - Consider the taxi agent making decisions from only the current camera snapshot

Model-based reflex agents



Model-based reflex agents remember state;
Have a model how the world works and keeps track of the part of the world it can'

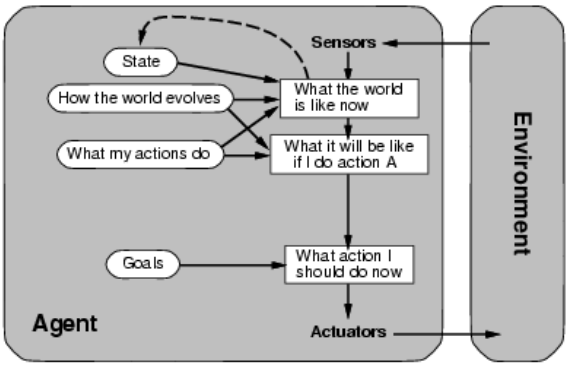
Model-based Reflex Agent

function Model-Based-Reflex-Agent(*percept*) **returns** an action
static: *rules*, a set of condition-action rules
state, a description of the current world state
action, the most recent action, initially none

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

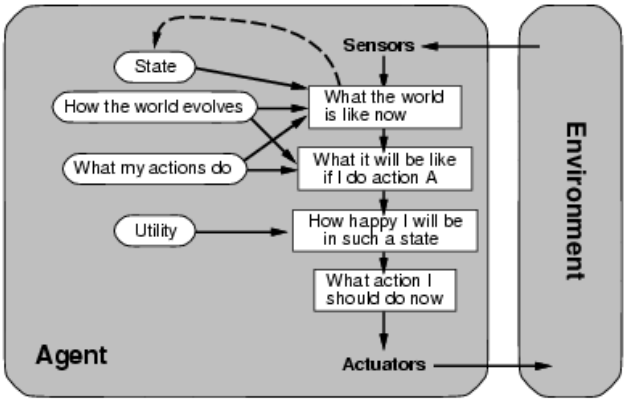
UPDATE-STATE is responsible for creating the new internal state description

Goal-based Agent



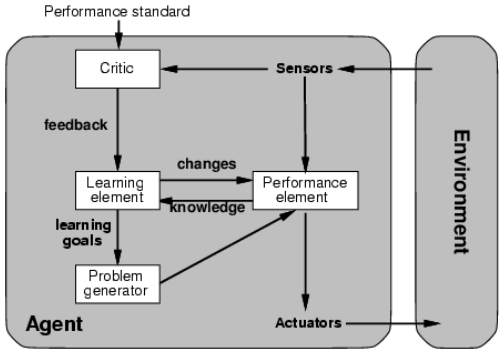
Just knowing state often not enough; needs a **goal**
e.g. taxi needs to know destination
Often requires **planning** and **search** to achieve the goal
Allows great flexibility in choosing actions to achieve goal

Utility-based agents



A utility function maps a state(s) to a number that describes the degree of happiness
Allows the agent to choose paths that may be better than others to achieve the goal

Learning agents



"Performance element" is essentially what we considered the entire agent
e.g. taxi skids on ice

Summary

Agents interact with environments through actuators and sensors

The agent function describes what the agent does in all circumstances

The performance measure evaluates the environment sequence

A perfectly rational agent maximizes expected performance

Agent programs implement (some) agent functions

PEAS descriptions define task environments

Environments are categorized along several dimensions:
observable? deterministic? episodic? static? discrete? single-agent?

Several basic agent architectures exist:
reflex, reflex with state, goal-based, utility-based