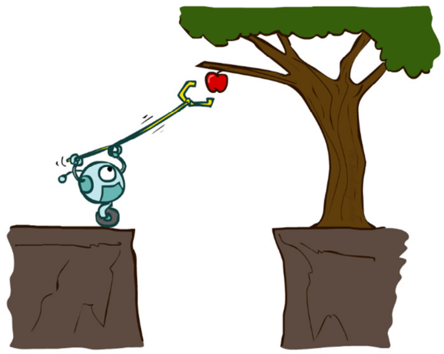
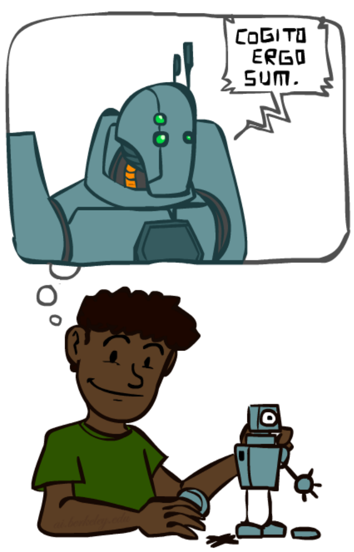
Artificial Intelligence
Chapter 2
Agents



Updates and Additions: Dr. Siamak Sarmady

By: Dan Klein and Pieter Abbeel
University of California, Berkeley

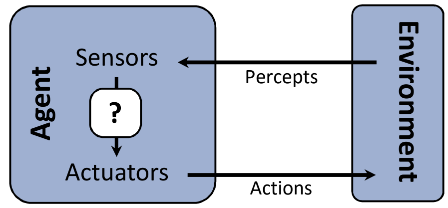
Designing Agents

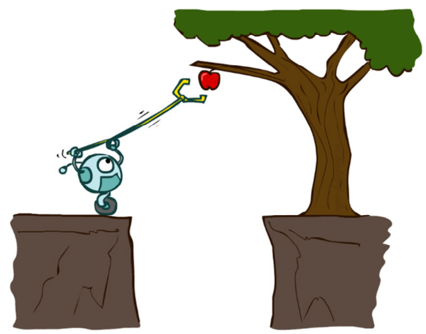


Demo: HISTORY – MT1950.wmv

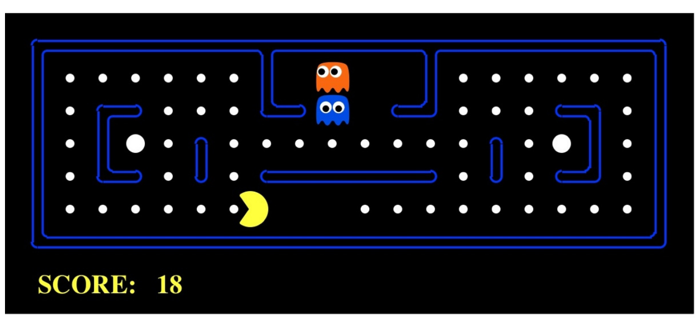
Designing Rational Agents

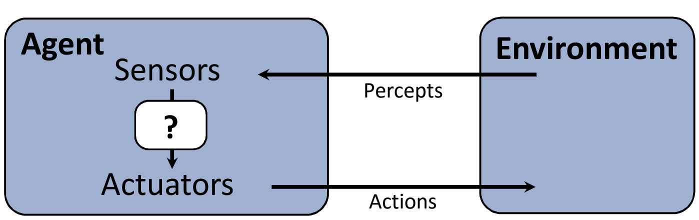
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.
- In short: An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that maximize its (expected) **utility**.
- Characteristics of the **percepts, environment,** and **action space** dictate techniques for selecting rational actions and also the complexity of an agent (thermostat, a complex robot)





Pac-Man as an Agent





Pac-Man is a registered trademark of Namco-Bandai Games, used here for educational purposes

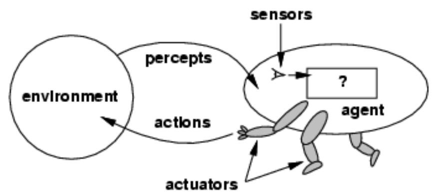
Demo1: pacman-l1.mp4 or L1D2

Pac-Man as an Agent

Example Agents

- **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

Agents and environments

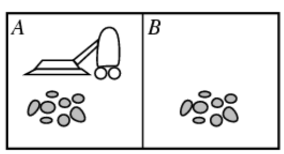


- The agent function maps from percept histories to actions:

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

- If we can specify the agent's choice of action for every possible percept sequence, then we have said more or less everything there is to say about the agent.

Vacuum-cleaner world



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

A vacuum-cleaner agent

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

```
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 function?
Can it be implemented in a small agent program?

Agents Architecture

- The agent function is an abstract mathematical description; the agent program is a concrete implementation, running on the agent architecture.

Agent = program + architecture

- The agent program runs on the physical architecture to produce f

Architecture = sensors + actuators

Rational agents

- 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.
- Aim: A rational agent wants to maximize the performance measure.

Rational agents

Assume an agent has committed an action. We want to asses whether it has been rational?

- Rationality depends on
  1. The performance measure that defines the criterion of success
  2. The agent's prior knowledge of the environment (i.e. sensor capabilities)
  3. The action's that agent can perform (i.e. actuator capabilities)
  4. The agent's percept sequence to date
- 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

- **Omniscience vs. rational:** Rationality is distinct from omniscience (all-knowing with infinite knowledge). An omniscience agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.*

- **Information gathering and exploration:** Agents can perform actions in order to modify future percepts so as to obtain useful information.

- **Autonomous agent:** An agent is autonomous if its behavior is determined by its own experience (with ability to learn and adapt) rathet than the knowledge of its designers

PEAS

- Before designing an agent, we must first specify the setting for the intelligent agent. These settings are sometimes called PEAS.

- PEAS:
  1. Performance measure (in addition to measuring success, it is used to improve the performance)
  2. Environment
  3. Actuators
  4. Sensors

PEAS

- 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, keyboard

PEAS

- Agent: Medical diagnosis system
  - **Performance measure:** Increase in patient health, 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

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 (and won't effect decisions of next episode).

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. multi-agent): An agent operating by itself in an environment.

known(vs unknown): the outcomes or outcomes probabilities for all actions are given.

Environment types

	Chess with a clock	Chess without a clock	Taxi driving
Fully observable	Yes	Yes	No
Deterministic	Strategic	Strategic	No
Episodic	No	No	No
Static	Semi	Yes	No
Discrete	Yes	Yes	No
Single agent	No	No	No

- The environment type largely determines the agent design
- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

Agent functions and programs

- An agent is completely specified by the agent function i.e. mapping percept sequences to actions
- An agent function (or a small equivalence class) is rational
- **Aim:** find a way to implement the rational agent function concisely
- Key challenge for AI is to find, how to write program that produce rational behavior from a small amount of code rather than from a large number of table entries.

Table-lookup agent

function TABLE-DRIVEN-AGENT(*percept*) **returns** *action*
static: *percepts*, a sequence, initially empty
table, a table, indexed by percept sequences, initially fully specified

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

Figure 2.5 An agent based on a prespecified lookup table. It keeps track of the percept sequence and just looks up the best action.

- Drawbacks:
  - Huge table ($\sum^T P^T$ entries)
  - Take a long time to build the table
  - No autonomy
  - Even with learning, need a long time to learn the table entries

Agent program for a vacuum-cleaner 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*

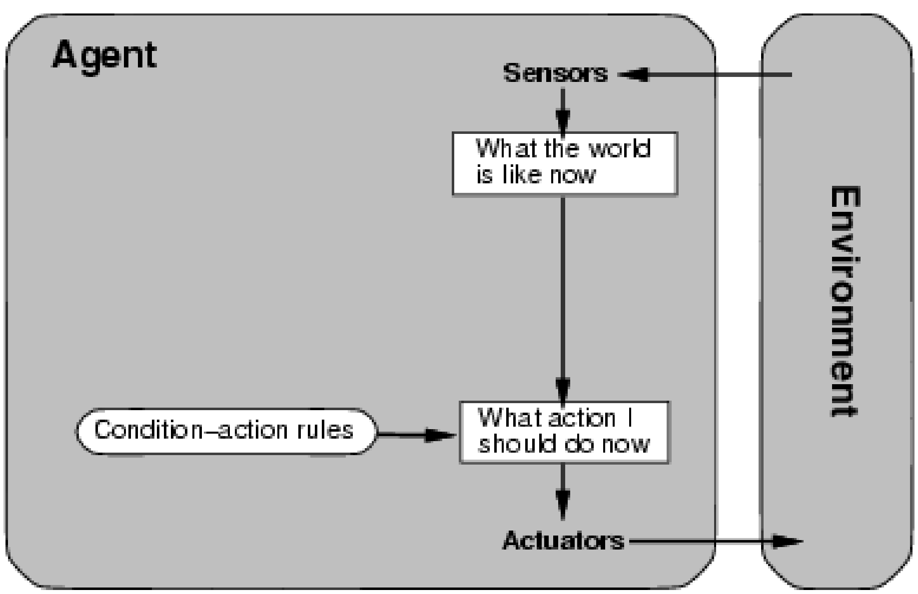
Agent types

- Four basic types in order of increasing generality:

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

- Learning Agents

Simple reflex agents



Simple reflex agents

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

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

Figure 2.8 A simple reflex agent. It works by finding a rule whose condition matches the current situation (as defined by the percept) and then doing the action associated with that rule.

- These agents select actions on the basis of only the current percept, *ignoring* the rest of the percept history (e.g. vacuum agent, because based on current location and whether there it contains dirt).

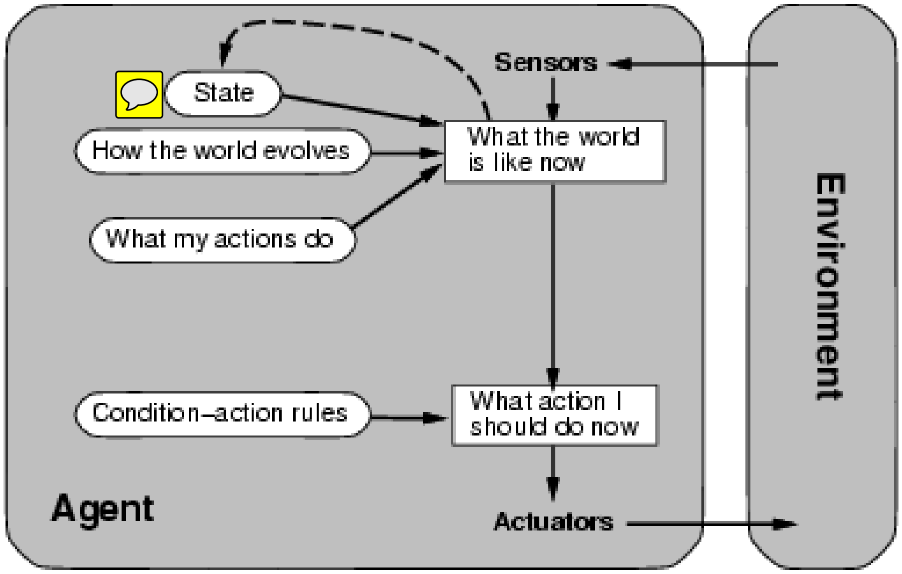
Simple reflex agents

- Vacuum agent has just 4 percepts in this way (down from 4^T).
- Condition-action rules (i.e. the first rule in the set that matches)

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

- Rectangles denote current internal state of the agent's decision process, the ovals represent the background information used in the process.
- Works only if the correct decision can be made on the basis of only current percept, that is only if the environment is fully observable.

Model-based reflex agents



Model-based reflex agents

- Keeps track of the part of the world it can see now i.e. maintains some sort of internal state that depends on the percept history (even though does not keep a percept history)

- Updating state information requires two kind of knowledge:
  - **First:** some info on how the world evolves independently of the agent, e.g. where will be the car nearby given we knew its location a while back
  - **Second:** how the agent's own actions affect the world, e.g. if the agent turns the steering wheel toward north, we will be 5 miles north in 5 minutes

Model-based reflex agents

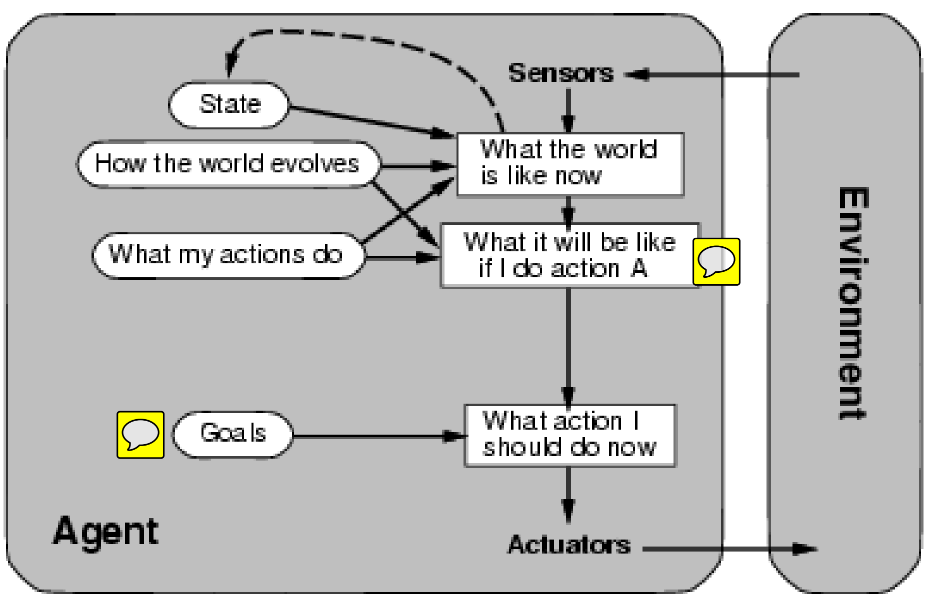
```
function REFLEX-AGENT-WITH-STATE(percept) returns action
  static: state, a description of the current world state
          rules, a set of condition-action rules

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

(state, action, percep

Figure 2.10 A reflex agent with internal state. It works by finding a rule whose condition matches the current situation (as defined by the percept and the stored internal state) and then doing the action associated with that rule.

- Current percept is combined with the old state to generate the updated description of the current state.
- Update-State takes into account the previous action (In fact, the new percept, gives insight about the effect of agent's action on the state of environment).

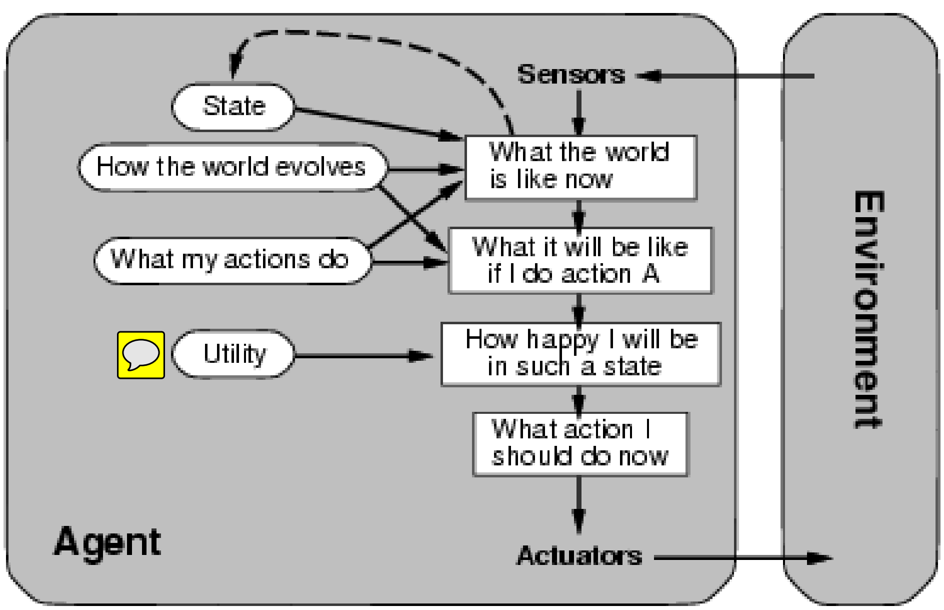
Goal-based agents



Goal Based Agent

- In addition to a current state description, the agent needs some sort of goal information that describes situations that are desirable. (e.g. for Taxi, the correct decision depends on where the taxi wants to go).
- Search and planning in the AI are devoted to find actions that achieve the agent's goals.
- Decision making of this kind is fundamentally different from rule based method. It involves the future, both "what will happen if I do such and such" and "Will that make me happy"?

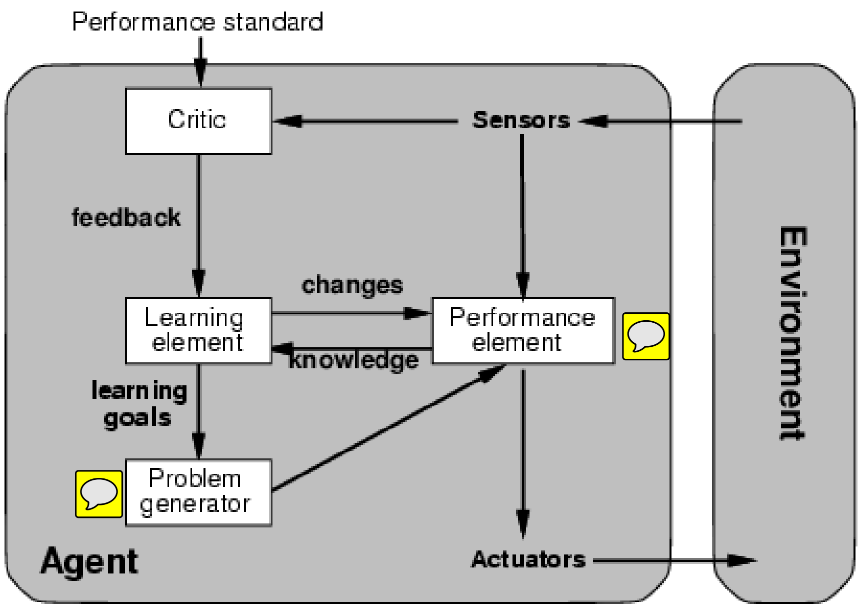
Utility-based agents



Utility Based Agent

- There are multiple action sequences that will get a taxi to destination (i.e. achieving goal) but some are quicker, safer, more reliable, or cheaper
- Goals just provide a crude binary distinction between "happy" and "unhappy" state. Utility function is a more general performance measure which produces a real number representing associated level of happiness.
- If one world state is preferred to another, then it has a higher utility for the agent.

Learning agents



Learning Agent

- In most areas of AI, it is the preferred method to build learning machines and then teach them.
- It allows the agent to operate in **initially** unknown environments and to become more competent than its initial knowledge alone might allow.
- **Performance (Execution) element:** takes the percepts and decides the action (the entire agent of previous types).
- **Learning element:** responsible for making improvements. Uses feedback from Critic to identify how the performance element should be modified to do better in the future. Its design depend very much on the design of performance element.

Learning Agent

- **Critic:** is necessary because the percepts themselves do not provide indication of agent's success. It is necessary that performance standard be fixed (outside the control of agent, so that agent cannot modify it to fit its behavior).
- **Problem Generator:** Responsible for suggesting exploratory actions that will lead to new and informative experiences.
- **Learning element:** can formulate a rule saying this was a bad action and it then installs the new rule in the performance element (modifies it).

Course Topics

- Part I: Making Decisions
  - Uninformed Search
  - Informed Search
  - Constraint satisfaction
  - Adversarial search
- Part II: Logic
  - Logical Agents
  - 1^{st} Order Logic
  - Resolution using 1^{st} Order Logic
- Additional
  - Machine Learning