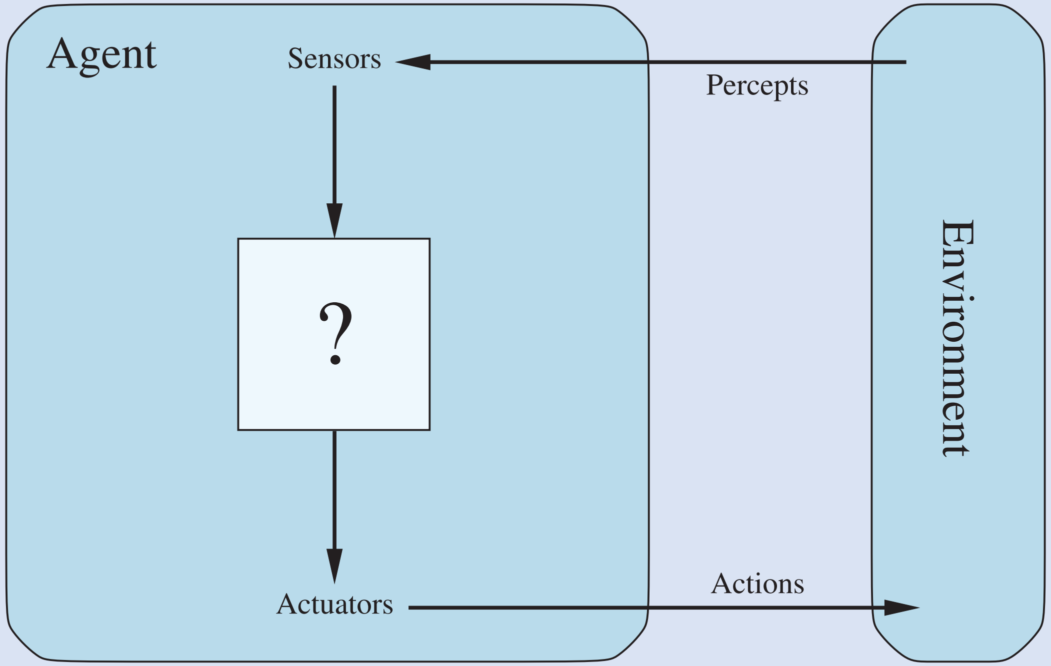
Artificial Intelligence

Intelligent Agents

Dr. Bilgin Avenoğlu

Agents and Environments

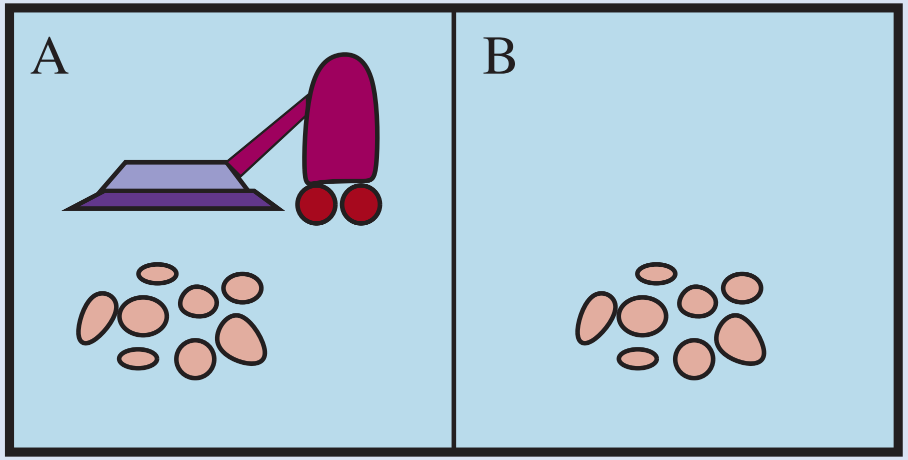
- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.



Agents interact with environments through sensors and actuators.

Example - Vacuum Agent

- The vacuum agent perceives which square it is in and whether there is dirt in the square.
- The agent starts in square A.
- The available actions are to
  - move to the right,
  - move to the left,
  - suck up the dirt,
  - do nothing.



- One very simple agent function is the following: if the current square is dirty, then suck; otherwise, move to the other square.

Intelligent?

- What is the right way to fill out the table?
- In other words, what makes an agent good or bad, intelligent or stupid?

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

Figure 2.3 Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2. The agent cleans the current square if it is dirty, otherwise it moves to the other square. Note that the table is of unbounded size unless there is a restriction on the length of possible percept sequences.

The Concept of Rationality

- A rational agent is one that does the right thing.
- Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

Performance Measures

- When an agent is plunked down 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.
- This notion of desirability is captured by a performance measure that evaluates any given sequence of environment states.
  - amount of dirt cleaned up in a single eight-hour shift.
    - a rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on.
  - reward the agent for having a clean floor.
    - one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated).

Rationality

- What is rational at any given time depends on four things:
  - 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.
- This leads to a 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

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known a priori but the dirt distribution and the initial location of the agent are not.
  - Clean squares stay clean and sucking cleans the current square. The Right and Left actions move the agent one square except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are Right, Left, and Suck.
- The agent correctly perceives its location and whether that location contains dirt.

- Under these circumstances the agent is indeed rational; its expected performance is at least as good as any other agent's.

Omniscience, Learning, and Autonomy

- An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.
  - information gathering—is an important part of rationality
    - exploration
- A rational agent should not only gather information but also to learn as much as possible from what it perceives.
- To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts and learning processes, we say that the agent lacks autonomy.

Example

- The female sphex will dig a burrow, go out and sting a caterpillar and drag it to the burrow, enter the burrow again to check all is well, drag the caterpillar inside, and lay its eggs.
- The caterpillar serves as a food source when the eggs hatch.
- If an entomologist moves the caterpillar a few inches away while the sphex is doing the check, it will revert to the "drag the caterpillar" step of its plan and will continue the plan without modification, re-checking the burrow, even after dozens of caterpillar-moving interventions.
- The sphex is unable to learn that its innate plan is failing, and thus will not change it.

Task Environment

- The performance measure, the environment, and the agent’s actuators and sensors.
  - We group all these under the heading of the task environment.
  - PEAS (Performance, Environment, Actuators, Sensors)

PEAS - automated taxi

- To design a rational agent, we must specify the task environment

- Consider, e.g., the task of designing an automated taxi:
  - Performance measure?? safety, destination, profits, legality, comfort, . . .
  - Environment?? US streets/freeways, traffic, pedestrians, weather, . . .
  - Actuators?? steering, accelerator, brake, horn, speaker/display, . . .
  - Sensors?? video, accelerometers, gauges, engine sensors, keyboard, GPS, . . .

PEAS - Internet shopping agent

- Performance measure?? price, quality, appropriateness, efficiency
- Environment?? current and future WWW sites, vendors, shippers
- Actuators?? display to user, follow URL, fill in form
- Sensors?? HTML pages (text, graphics, scripts)

Properties of Task Environments

- Fully observable vs. partially observable
  - noisy and inaccurate sensors, unknown maps etc.
- Single-agent vs. multiagent
  - a crossword puzzle - single
  - playing chess - two-agents
- Competitive vs. cooperative
- Deterministic vs. nondeterministic
  - if the next state of the environment is completely determined - deterministic.
  - taxi driving - nondeterministic,
  - vacuum world - deterministic
- Episodic vs. sequential
  - the next episode does not depend on the actions taken in previous episodes
  - defective parts on an assembly - episodic
  - chess and taxi driving are sequential
- Static vs. dynamic
  - taxi driving is clearly dynamic
  - crossword puzzles are static
- Discrete vs. continuous
  - the speed and location of the taxi
- Known vs. unknown
  - in a known environment, the outcomes for all actions are given.

Environment Types

- The environment type largely determines the agent design

- The real world is (of course) partially observable, stochastic, sequential, dynamic, continuous, multi-agent

	Solitaire	Backgammon	Internet shopping	Taxi
Observable??	Yes	Yes	No	No
Deterministic??	Yes	No	Partly	No
Episodic??	No	No	No	No
Static??	Yes	Semi	Semi	No
Discrete??	Yes	Yes	Yes	No
Single-agent??	Yes	No	Yes (except auctions)	No

Agent Types

- Four basic types in order of increasing generality:
  - simple reflex agents
  - reflex agents with state
  - goal-based agents
  - utility-based agents

- All these can be turned into learning agents

Simple Reflex Agents

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

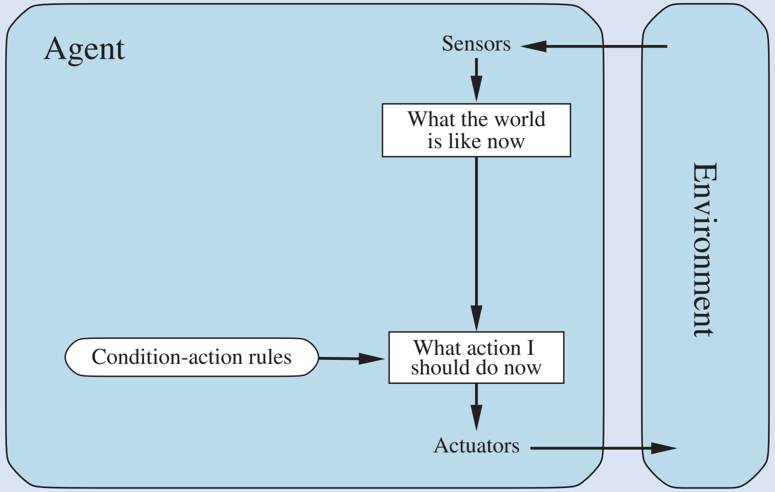


Figure 2.9 Schematic diagram of a simple reflex agent. We use rectangles to denote the current internal state of the agent's decision process, and ovals to represent the background information used in the process.

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

```
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*

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

Model-based Reflex Agents

- To handle partial observability - keep track of the part of the world that the agent 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.

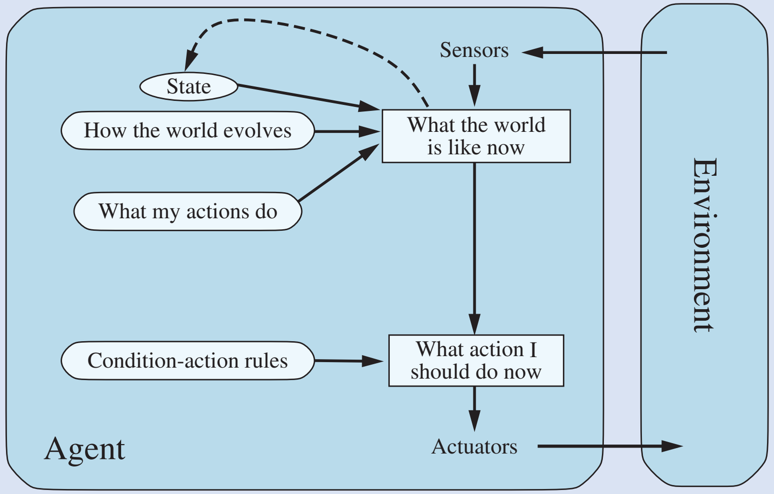


Figure 2.11 A model-based reflex agent.

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              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
```

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.

Goal-based Agents

- As well as a current state, the agent needs some sort of goal information that describes situations that are desirable
  - for example, being at a particular destination.
- The agent program can combine the goal with the model to choose actions that achieve the goal.
- Search (Chapters 3, 4, and 6) and planning (Chapter 11) are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

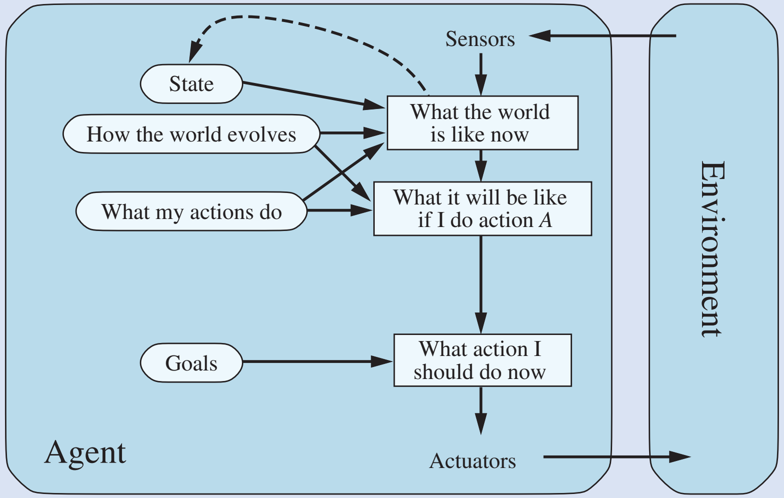


Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

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, but some are quicker, safer, more reliable, or cheaper than others.

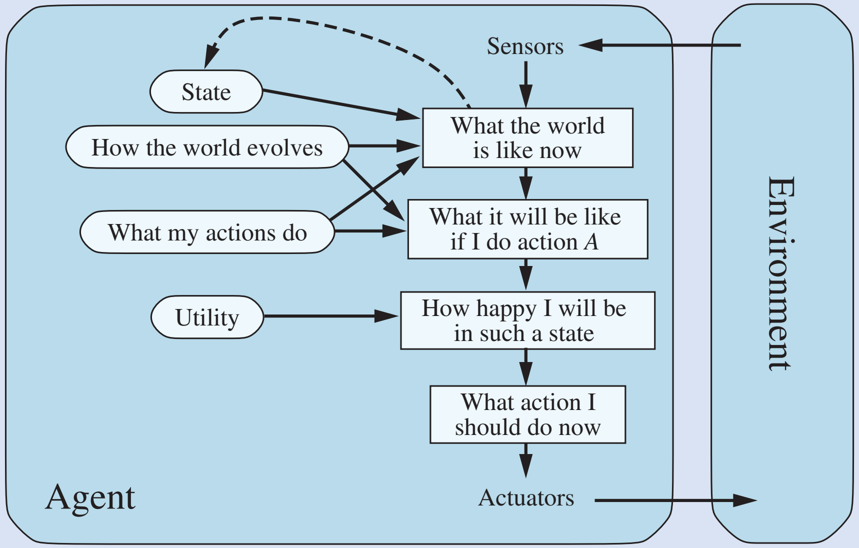


Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome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

Learning Agents

- Learning element is responsible for making improvements.
- The performance element, or agent: takes percepts and decides on actions.
- The learning element
  - uses feedback from the critic on how the agent is doing
  - determines how the performance element should be modified to do better in the future.

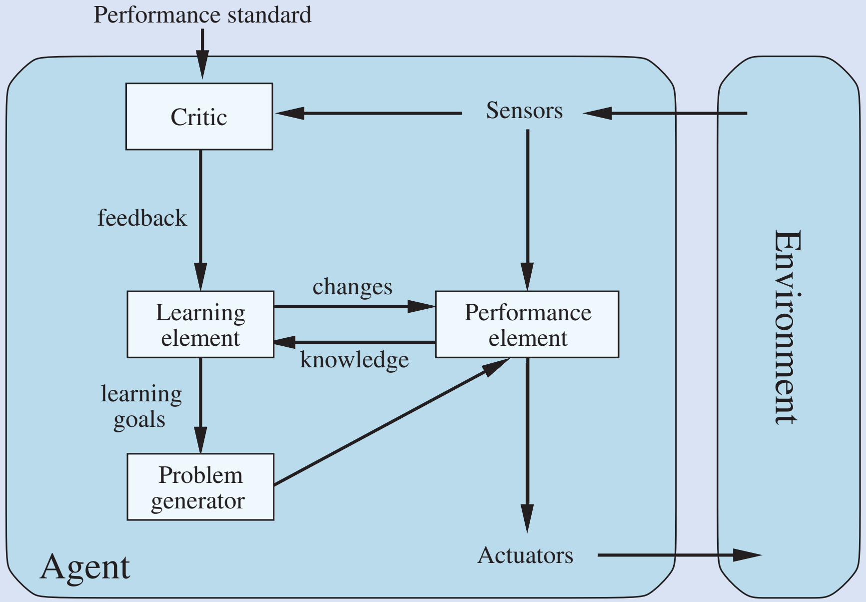


Figure 2.15 A general learning agent. The "performance element" box represents what we have previously considered to be the whole agent program. Now, the "learning element" box gets to modify that program to improve its performance.

- Problem generator is responsible for suggesting actions that will lead to new and informative experiences.

How the Components of Agent Programs Work?

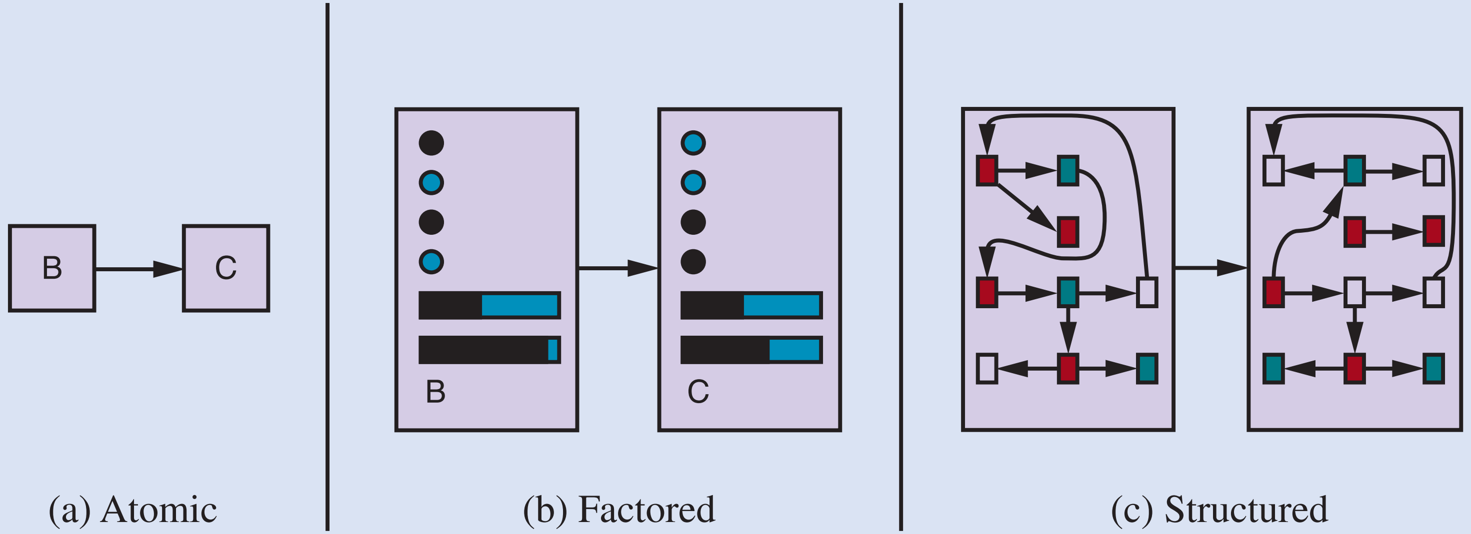


Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

How the Components of Agent Programs Work?

- In an atomic representation each state of the world is indivisible - it has no internal structure.
  - Search and game-playing (Chapters 3, 4, and 6),
  - Hidden Markov models (Chapter 14),
  - Markov decision processes (Chapter 16).
- A factored representation splits up each state into a fixed set of variables or attributes, each of which can have a value.
  - Constraint satisfaction algorithms (Chapter 5),
  - Propositional logic (Chapter 7),
  - Planning (Chapter 11),
  - Bayesian networks (Chapters 12, 13, 14, 15, and 18),
  - Machine learning algorithms.
- In structured representation, objects and their various and varying relationships can be described explicitly.
  - relational databases and first-order logic (Chapters 8, 9, and 10),
  - first-order probability models (Chapter 18),
  - natural language understanding (Chapters 24 and 25).
    - In fact, much of what humans express in natural language concerns objects and their relationships.

The End!