What did we learn?

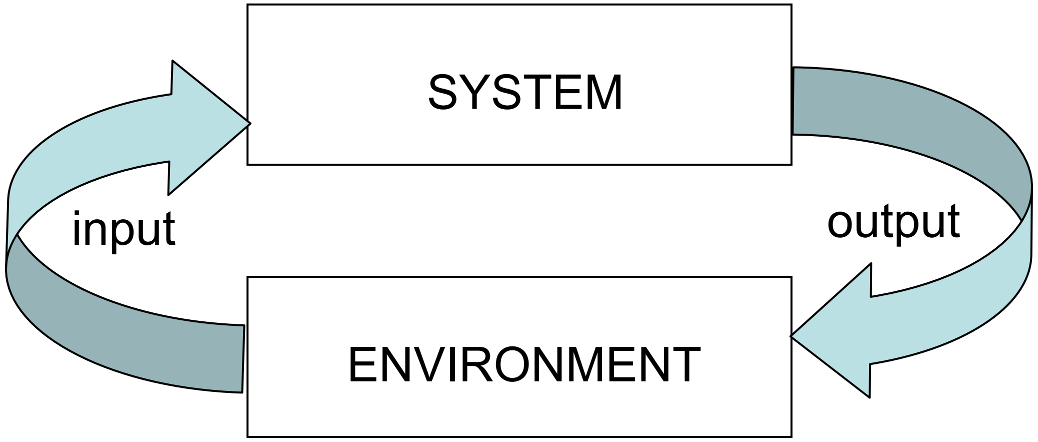
Views of AI fall into four categories:□

Thinking humanly	Thinking rationally
Acting humanly	**Acting rationally**

The textbook advocates "acting rationally"□

What is an Agent?

- The main point about agents is they are *autonomous*: capable of acting independently, exhibiting control over their internal state
- Thus: *an agent is a computer system capable of* autonomous action *in some environment in order to meet its* design objectives



What is an Agent?

- Trivial (non-interesting) agents:
  - thermostat
  - UNIX daemon (e.g., biff)
- *An* intelligent agent *is a computer system capable of* flexible *autonomous action in some environment*
- By *flexible*, we mean:
  - *reactive*
  - *pro-active*
  - *social*

Reactivity

- If a program's environment is guaranteed to be fixed, the program need never worry about its own success or failure – program just executes blindly
  - Example of fixed environment: compiler
- The real world is not like that: things change, information is incomplete. Many (most?) interesting environments are *dynamic*
- Software is hard to build for dynamic domains: program must take into account possibility of failure – ask itself whether it is worth executing!
- A *reactive* system is one that maintains an ongoing interaction with its environment, and responds to changes that occur in it (in time for the response to be useful)

Proactiveness

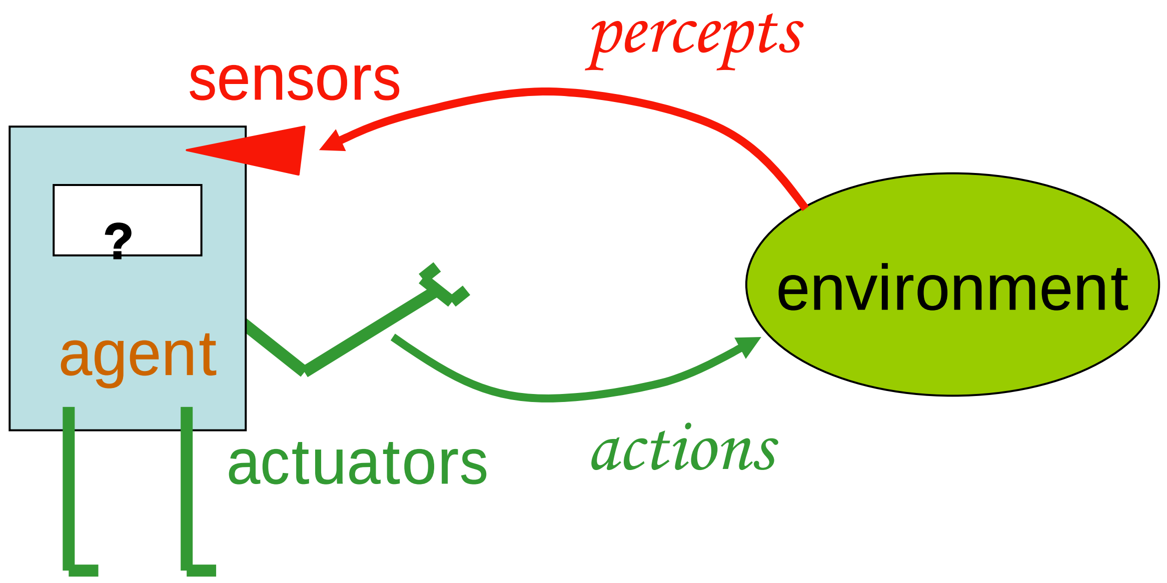
- Reacting to an environment is easy (e.g., stimulus $\rightarrow$ response rules)
- But we generally want agents to *do things for us*
- Hence *goal directed behavior*
- Pro-activeness = generating and attempting to achieve goals; not driven solely by events; taking the initiative
- Recognizing opportunities

Balancing Reactive and Goal-Oriented Behavior

- We want our agents to be reactive, responding to changing conditions in an appropriate (timely) fashion
- We want our agents to systematically work towards long-term goals
- These two considerations can be at odds with one another
- Designing an agent that can balance the two remains an open research problem

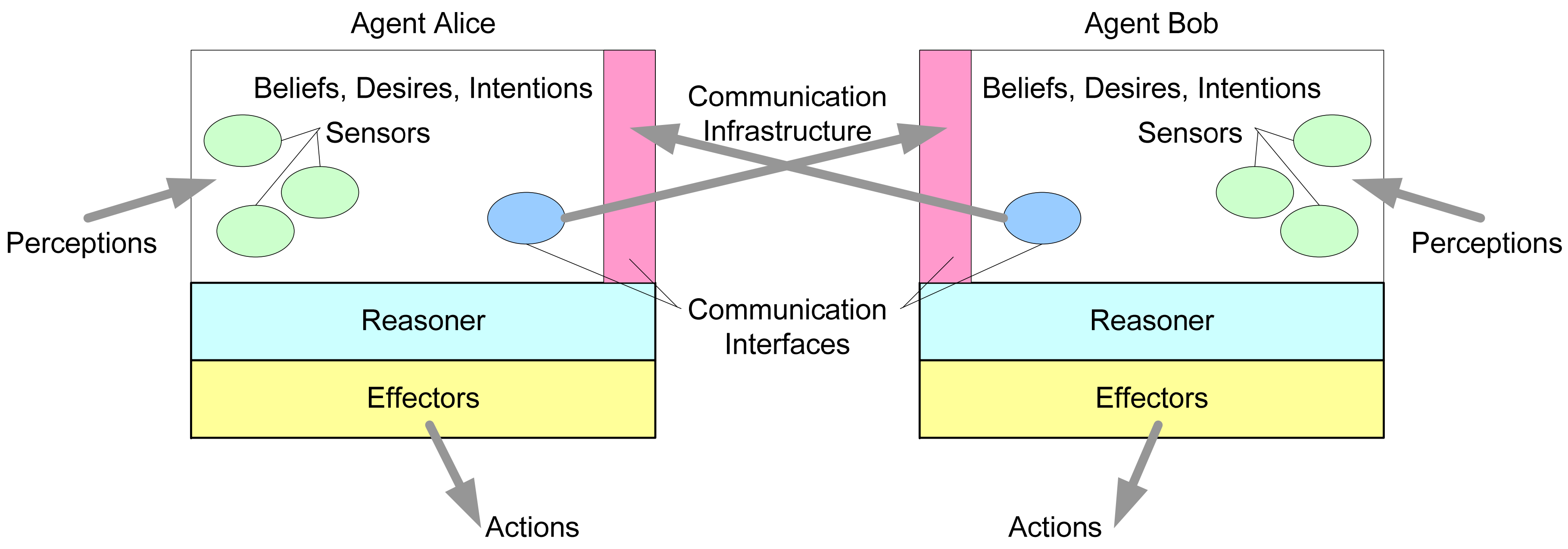
What is an Agent?

An **intelligent agent** perceives its **environment** via **sensors** and acts rationally upon that environment with its **actuators**.



Cognitive Architecture for an Agent

Called a BDI (beliefs, desires, intentions) architecture

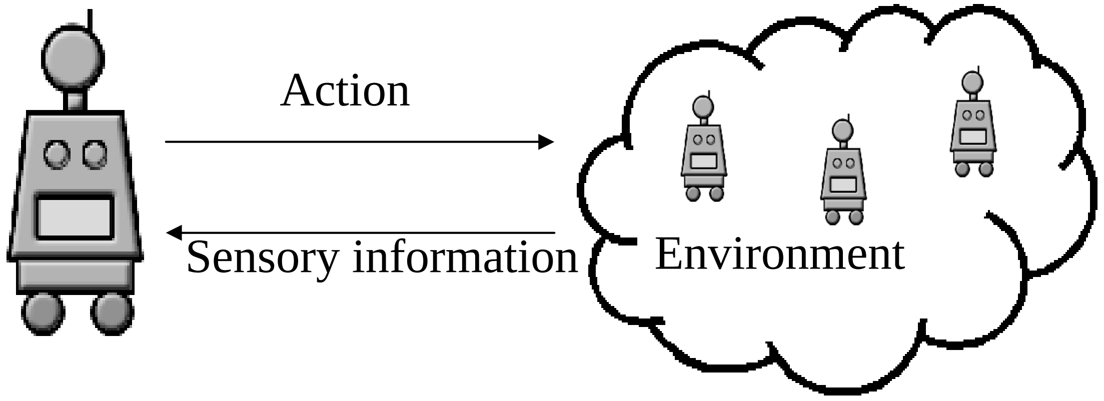


Like the reactive architecture at a coarse level, but with two differences:

- Cognitive representations
- Deeper reasoning based on the above representations

Making decisions

- We require software agents to whom complex tasks and goals can be delegated
- Agents should be smart so that they can make decisions and take actions to successfully complete tasks and goals
- Endowing the agent with the capability to make good decisions is a nontrivial issue



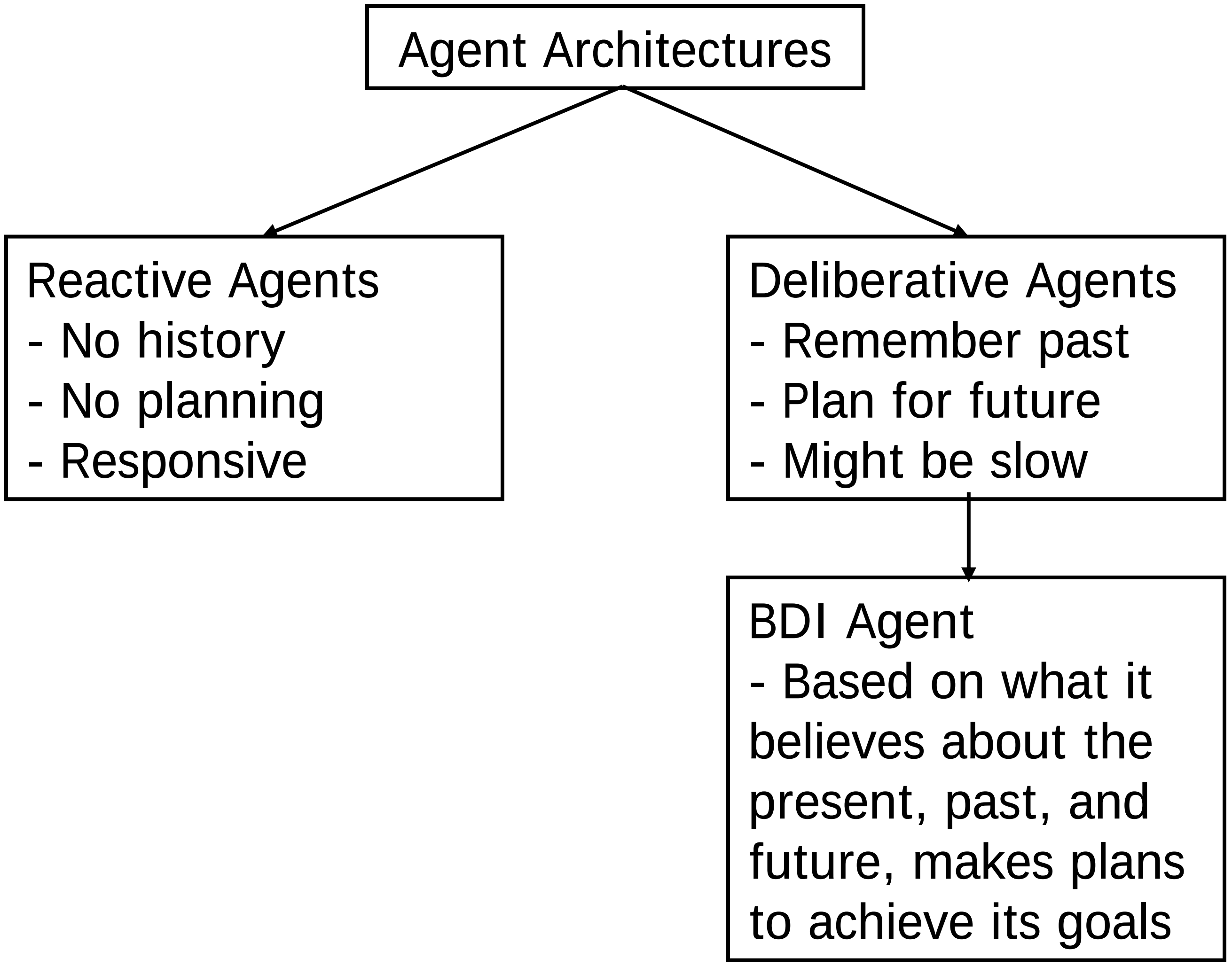
Structure of Agents

Agent = Architecture + Program

Computing device Running Agent Program, with sensors & actuators

Implements Agent Function, performs mapping of percepts to actions

Architectural Types

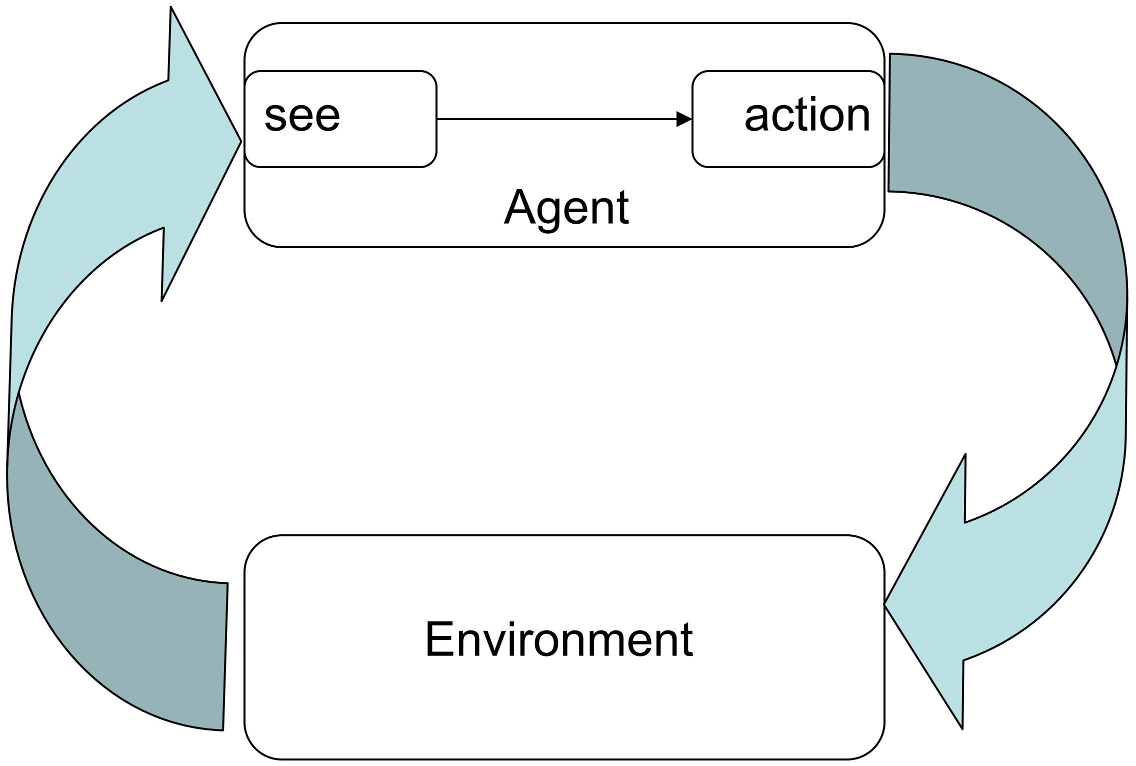


Agent Programs

- Kinds of Agent Programs
  - Simple Reflex Agents
  - Model-based Reflex Agents
  - Goal Based Reflex Agents
  - Utility-based Reflex Agents

Perception

- Now introduce *perception* system:



Perception

- The *see* function is the agent's ability to observe its environment, whereas the *action* function represents the agent's decision making process
- *Output* of the *see* function is a *percept*:

$$see : E \rightarrow Per$$

which maps environment states to percepts, and *action* is now a function

$$action : Per^* \rightarrow A$$

which maps sequences of percepts to actions

A simple view of an agent

- Environment states $S=\{s_1, s_2, \ldots\}$
- Perception *see*:$S \rightarrow P$
- An agent has an internal state (*IS*) which is updated by percepts:

  $next: IS \times P \rightarrow IS$
- An agent can choose an action from a set $A=\{a_1, a_2, \ldots\}$:

  $action: IS \rightarrow A$
- The effects of an agent's actions are captured via the function *do*:

  $do: A \times S \rightarrow S$

Structure of an Intelligent Agent

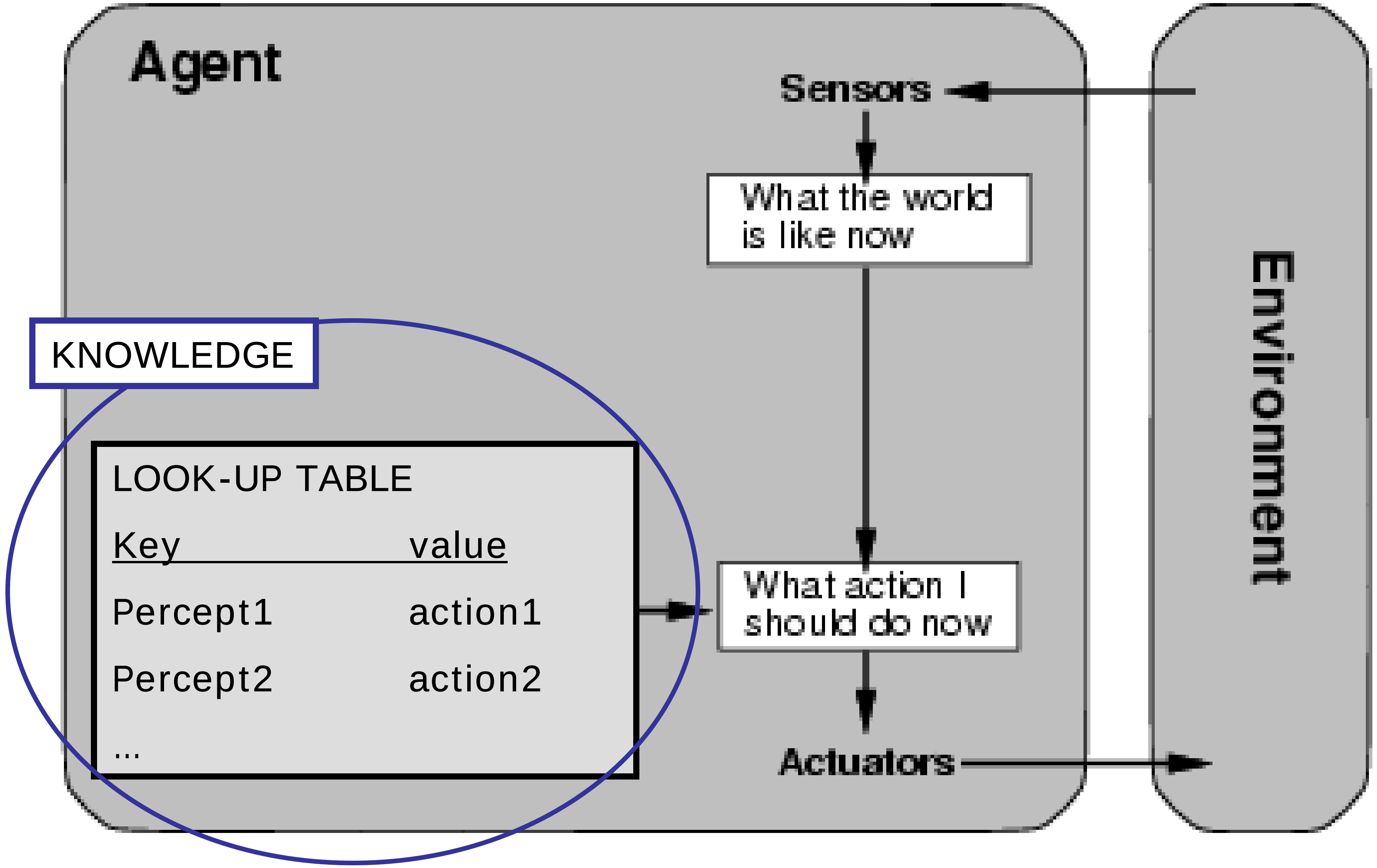
- All agents have the same basic structure:
  - accept percepts from environment
  - generate actions
- A Skeleton Agent Program:

```
function Skeleton-Agent(percept) returns action
  static: memory, the agent's memory of the world

  memory ← Update-Memory(memory, percept)
  action ← Choose-Best-Action(memory)
  memory ← Update-Memory(memory, action)
  return action
```

- Observations:
  - agent may or may not build percept sequence in memory (depends on domain)
  - performance measure is not part of the agent; it is applied externally to judge the success of the agent

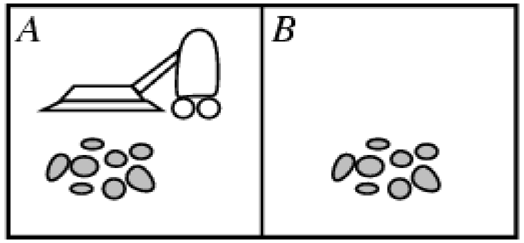
Table-driven agents (revised from R&N)



LOOK-UP TABLE

Key	value
Percept1	action1
Percept2	action2
...	

Example: Vacuum Cleaner Agent



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

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

Looking Up the Answer?

- A Template for a Table-Driven 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
```

- Why can't we just look up the answers?
  - The disadvantages of this architecture
    - infeasibility (excessive size)
    - lack of adaptiveness
  - How big would the table have to be?
  - Could the agent ever learn from its mistakes?
  - Where should the table come from in the first place?

Agent Types

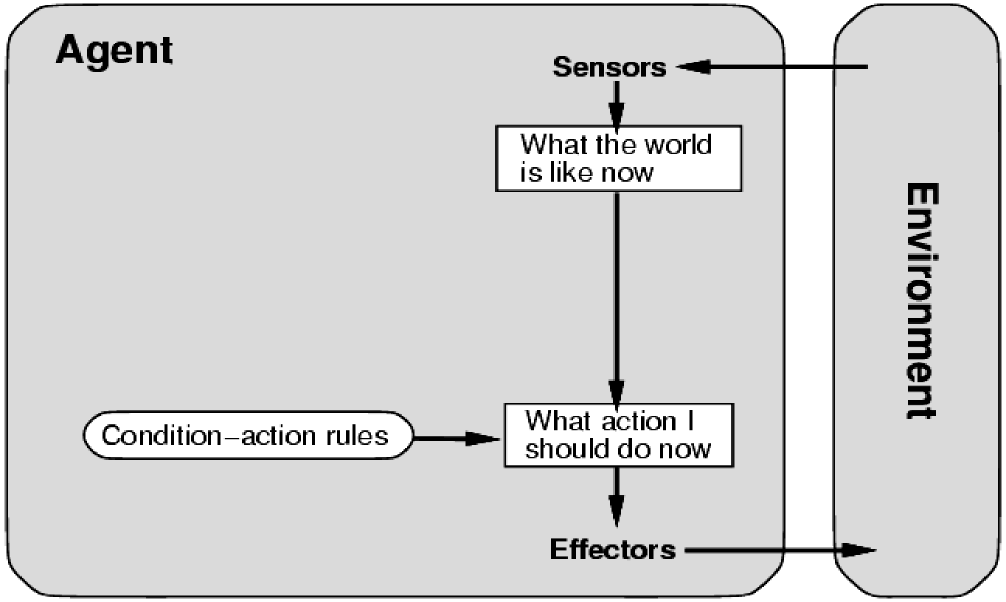
- Simple reflex agents
  - are based on condition-action rules and implemented with an appropriate production system. They are stateless devices which do not have memory of past world states.
- Reflex Agents with memory (Model-Based)
  - have internal state which is used to keep track of past states of the world.
- Agents with goals
  - are agents which in addition to state information have a kind of goal information which describes desirable situations. Agents of this kind take future events into consideration.
- Utility-based agents
  - base their decision on classic axiomatic utility-theory

A Simple Reflex Agent

- We can summarize part of the table by formulating commonly occurring patterns as condition-action rules:
- Example:

  if ***car-in-front-brakes***

  then ***initiate braking***
- Agent works by finding a rule whose condition matches the current situation

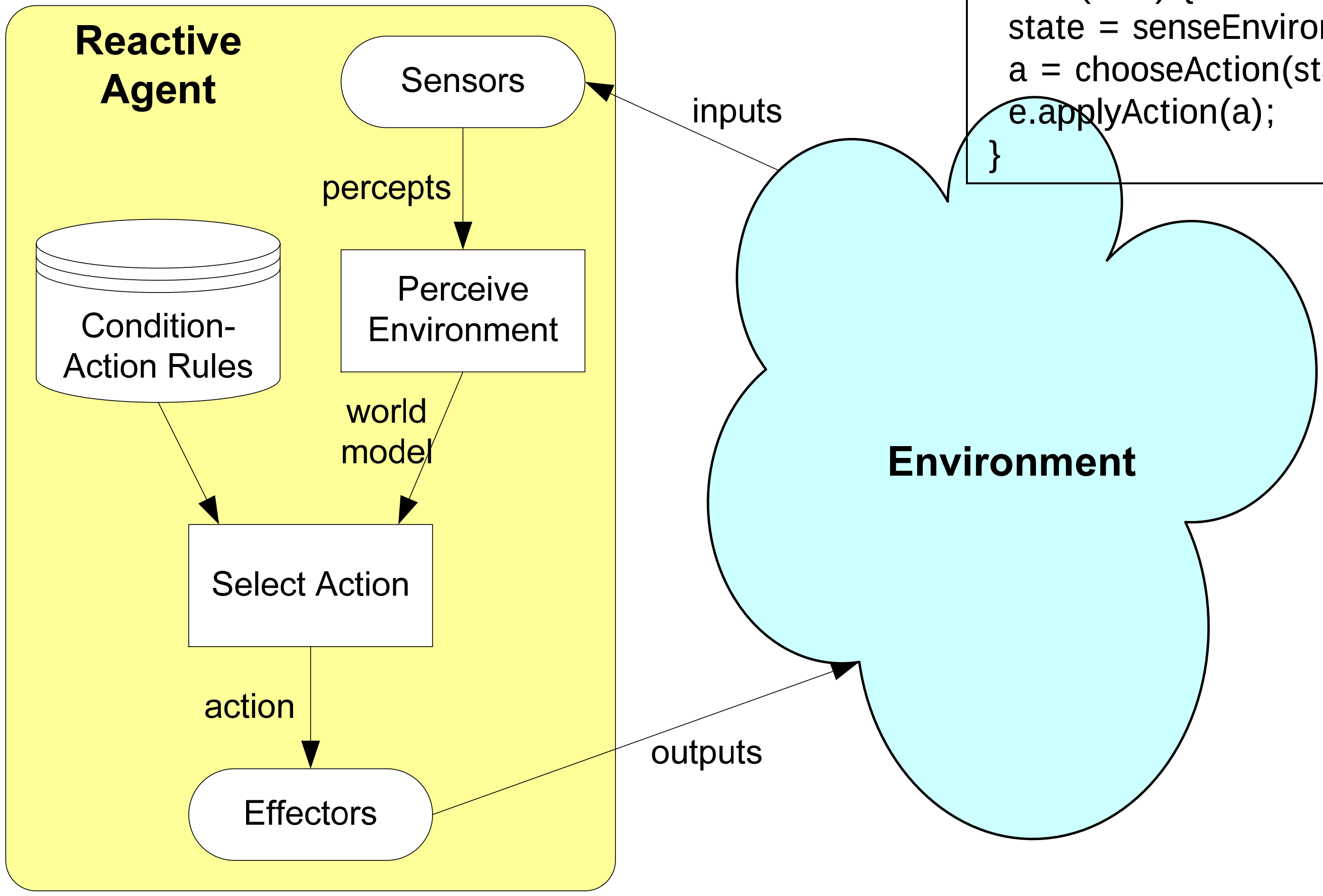


rectangles ← the current internal state; Ovals ← background information

```
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
```

A Reactive Agent in an Environment



```
Environment e;
RuleSet r;
while (true) {
  state = senseEnvironment(e);
  a = chooseAction(state, r);
  e.applyAction(a);
}
```

What is an Intelligent Agent

- Rationality depends on
  - the performance measure that defines degree of success
  - the percept sequence - everything the agent has perceived so far
  - what the agent know about its environment
  - the actions that the agent can perform

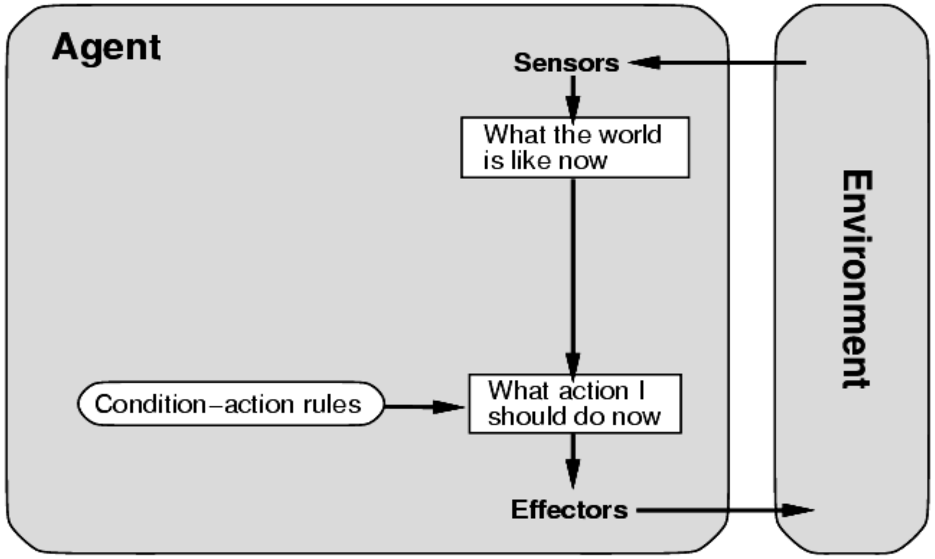
- Agent Function (percepts ==> actions)
  - Maps from percept histories to actions $f: \mathcal{P}^* \rightarrow \mathcal{A}$
  - The agent program runs on the physical architecture to produce the function *f*
  - agent = architecture + program

```
Action := Function(Percept Sequence)
If (Percept Sequence) then do Action
```

- Example: A Simple Agent Function for Vacuum World

```
If (current square is dirty) then suck
Else move to adjacent square
```

Example: Simple Reflex Vacuum 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 $Left$
```

Simple Reflex Agents: Remarks

- Considers only the current percept, ignores rest of 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 return Left
```

- Condition-action rules encoded
  - If car-in-front-is-braking then initiate-braking

```
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
```

rule-based systems

But, this only works if the current percept is sufficient for making the correct decision!

Simple reflex agents

Act only on the basis of the current percept.

The agent function is based on the

condition-action rule: condition $\Rightarrow$ action

Limited functionality:

Work well only when

- the environment is fully observable and
- the condition-action rules have predicted all necessary actions.