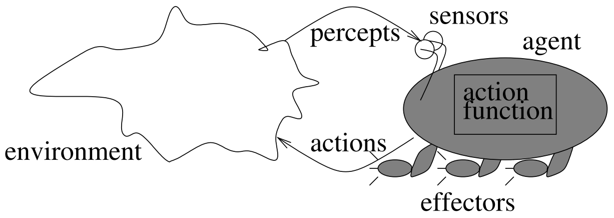
Agents

1 Sense, Plan, Act

The word ‘agent’ is heavily used in AI and Computer Science nowadays. There’s an attempt here to make sense of the multitude of uses.

> *“An* agent *is anything that can be viewed as* perceiving *its* environment *through* sensors *and* acting *upon that environment through* effectors.*”*



At some level of abstraction, agents execute a *sense/plan/act cycle*:

- *Sense:* Use sensors to find things out about the environment (including the effects of previous actions)
- *Plan:* Decide on the next action(s)
- *Act:* Use effectors to carry out the chosen action(s)

In AI, our focus is the *Plan* phase.

The task of the Plan phase is to implement an *action function* that maps

- from *percept sequences* (the sequence of things the agent has perceived so far)
- to the actions the agent can perform (to give the next action(s))

2 Intelligent Agents

It is really the *Plan* phase that determines whether we have an *intelligent agent*:

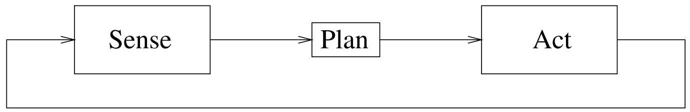
- *Rationality:* whether the action that the agent chooses is the one that it expects will maximize success;
- *Autonomy:* whether the agent chooses its actions without the direct intervention of controlling agents.

3 Different Types of Agent

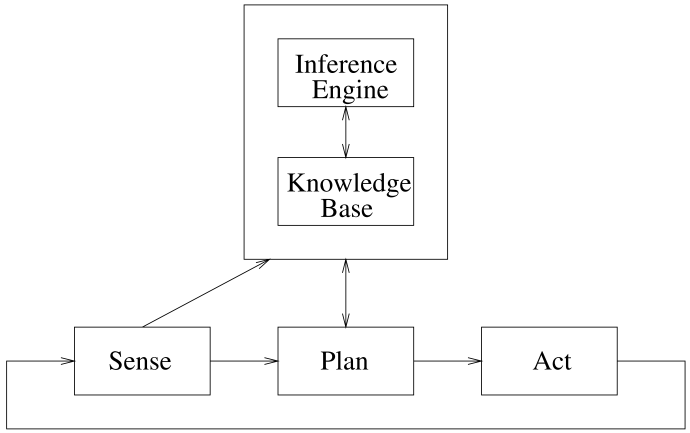
There are many ways to subcategorise intelligent agents.

3.1 Reactive Agents and Deliberative Agents

One distinction is whether an agent is purely reactive, or whether it also deliberates. A reactive agent is one that chooses its actions on the basis of immediate stimuli only. In other words, its next action is chosen solely using what its sensors tell it the world is like now. It doesn’t use any memory to keep track of what the world is like, and it does not think ahead. Reactive agents therefore have very simple *Plan* phases. Despite this, reactive agents can exhibit surprisingly sophisticated behaviour, although it is questionable whether a purely reactive agent could ever qualify as intelligent.



A deliberative agent, by contrast, will think ahead. It keeps a mental model of the world, and it ‘simulates’ the effects of actions on that mental model. It will often explore alternative ‘simulations’. It chooses its actions on the basis of the outcomes of these ‘simulations’: it can choose the actions that best achieve its goals or that bring greatest utility. It needs a memory and a reasoning capability, hence its *Plan* phase is much more complicated.



Intelligent agents, such as humans, are hybrids that combine reactive and deliberative capabilities.

3.2 Robots and Softbots

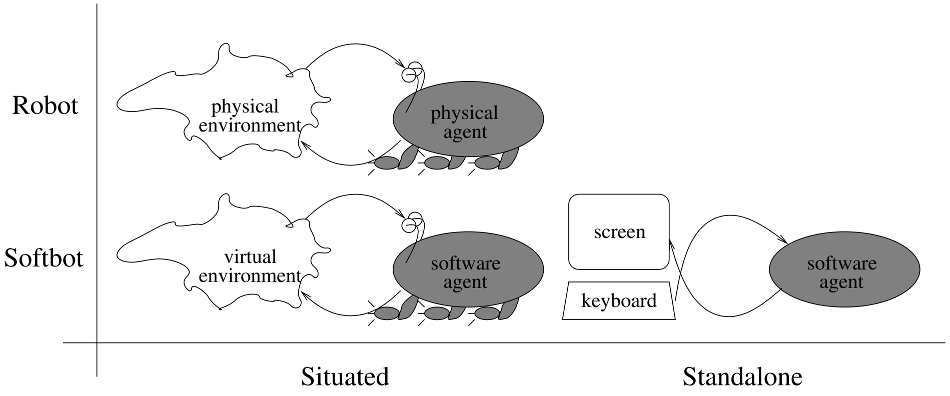
There are at least two other ways of classifying intelligent agents. The first is whether they have physical bodies (robots) or whether they are purely software agents (softbots). The second is whether they exist in some kind of world (situated) or not (standalone).

Robots *Embodied* agents, *situated* in physical environments
E.g. planetary rovers, warehouse part pickers

Softbots (Also known as *software agents*) *Situated* in virtual environments, e.g. the Internet, virtual realities

E.g. WWW bargain hunting agent, diary management agent As a special case, some softbots receive only symbolic input (e.g. from the keyboard) and their actions are limited to the production of symbolic output (e.g. screen output)

E.g. theorem-provers, medical consultancy systems, careers advisory systems



3.3 Agents in Computer Science

Now is as good a time as any to point out four of the many other ways that the word ‘agent’ is being used.

- Computer scientists working in the area of distributed systems talk of ‘agents’ and especially ‘mobile agents’. They’re referring to programs that can move around a computer network and can execute on a variety of different host platforms in that network. Agents of this kind might be intelligent agents, but they need not be.
- In the development of programming languages, people now talk of ‘agent-based languages’. These are like object-oriented languages, but the objects execute concurrently, and are called agents. Agents of this kind might be intelligent agents, but they need not be.
- People who are in the business of marketing computer applications are now using the word ‘agent’ a lot. There’s talk of email filtering agents, bargain hunting agents, deal-making agents, etc. What these ‘agents’ have in common is that they are computer applications in which some task is delegated to the agent, and done on behalf of a human user. Agents of this kind are often intelligent agents, but they need not be.
- In some theories of intelligence, minds are made up of many small processes. These processes are referred to as agents. Each of these agents is mindless (nonintelligent). But the mind is a society of these agents, and intelligence emerges from the workings of this society.

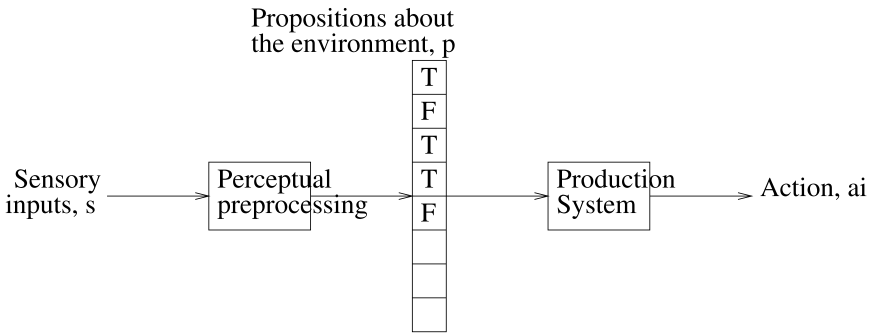
4 Building Reactive Agents using Production Systems

Let’s start seeing how to build intelligent agents, and let’s start with the architecturally simplest approaches. For the next few lectures, we’ll look at agents that are purely reactive.

One way of building such an agent is to use a *production system*.

Such an agent has sensory inputs $\mathbf{s} = \langle s_1, \ldots, s_n \rangle$. It applies some processing to these to then produce a vector $\mathbf{p} = \langle p_1, \ldots, p_i, \ldots, p_m \rangle$. For the moment, we will assume that each p_i is a Boolean value, either **true** or **false** . Each p_i represents some simple proposition about the environment, e.g. that there is an obstacle directly in front of the agent, or that the agent is in a corner. Finally, the agent has a vector of actions that it can choose to execute, $\mathbf{a} = \langle a_1, \ldots, a_l \rangle$.

The agent then needs a way of implementing its action function. This is where we use the production system. It will choose which action to execute on the basis of vector $\mathbf{p}$.



A production system comprises a list of *rules*, for which there is a variety of terminology: production rules, productions, condition-action rules, situation-action rules, stimulus-response rules, and SR rules. Each rule is of the form

$$\textbf{if} \quad c_i \quad \textbf{then} \quad a_i$$

where c_i is a *condition* and a_i is one of the agent’s possible actions. The conditions will be Boolean-functions of the propositions in $\mathbf{p}$. A convenient way to write the conditions is to write them using the operators of Boolean algebra, e.g. conjunction ($\wedge$), disjunction ($\vee$), negation ($\neg$), etc.

The agent will execute the action of one of the rules whose condition evaluates to **true** . Of course, in general, several rules will have true conditions. We refer to these as the *conflict set*. We will need a *conflict resolution strategy* to pick one of these rules.

In summary, if `rules` is the list of rules, then our agent does the following at each moment in time:

```
sense();
preprocess();
conflictSet = rules.findAllMatches();
rule = conflictSet.chooseARule();
action = rule.getAction();
action.execute(this);
```

There are many possible conflict resolution strategies, including:

- Choose the *first* rule that matches.
- Assign priorities to the rules. Choose the rule from the conflict set that has *highest priority*.

- Choose the *most specific* rule. Assuming rule conditions are conjunctions of propsoitions, this is the rule in the conflict set that has the longest condition.
- Choose a rule that has not been used before or the one that has been used *least recently*.

We will only consider production systems where the conflict resolution strategy is to execute the action of the *first* rule whose condition is **true** . This gives us a simplification of our algorithm:

```
sense();
preprocess();
rule = rules.findFirstMatch();
action = rule.getAction();
action.execute(this);
```

By way of a simple example, consider a robot who lives in a room in which there is a light source. The robot has two light sensors on its front: a left sensor and a right sensor. Each of these can obtain a real number to signify the amount of light being received. The robot is capable of three actions: TURN_LEFT, TURN_RIGHT and MOVE. These actions are turning $45°$ to the left, turning $45°$ to the right, and moving one pace forward, respectively.

So, vector $\mathbf{s}$ is of length two: s_0 says how much light the left sensor is receiving; s_1 says how much light the right sensor is receiving.

Vector $\mathbf{p}$ is also of length two. p_0 will be **true** iff $s_0 > s_1$; p_1 will be **true** iff $s_1 > s_0$. (If p_0 and p_1 are both **false** , then the two sensors must be receiving the same amount of light.)

The production system is:

if p_0 **then** TURN_LEFT
if p_1 **then** TURN_RIGHT
if $\neg p_0 \wedge \neg p_1$ **then** MOVE

Question: *What will be the behaviour of this robot?*