Artificial Intelligence I

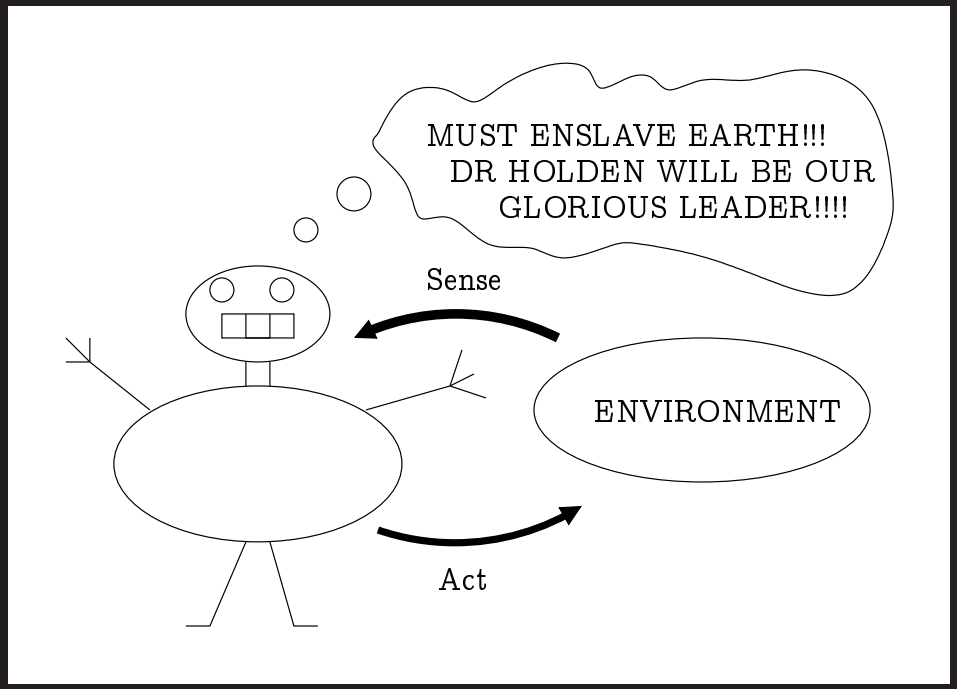
Dr Sean Holden

An introduction to *Agents*

Agents

There are many different definitions for the term *agent* within AI.

Allow me to introduce EVIL ROBOT.



We will use the following simple definition: *an agent is any device that can sense and act upon its environment*.

Agents

This definition can be very widely applied: to humans, robots, pieces of software, and so on.

We are taking quite an *applied* perspective. We want to *make things* rather than *copy humans*, so to be scientific there are some issues to be addressed:

- How can we judge an agent's performance?
- How can an agent's *environment* affect its design?
- Are there sensible ways in which to think about the *structure* of an agent?

Recall that we are interested in devices that *act rationally*, where 'rational' means doing the *correct thing* under *given circumstances*.

Reading: Russell and Norvig, chapter 2.

Measuring performance

How can we judge an agent's performance? Any measure of performance is likely to be *problem-specific*.

Example: For a chess playing agent, we might use its rating.

Example: For a mail-filtering agent, we might devise a measure of how well it blocks spam, but allows interesting email to be read.

Example: For a car cleaning robot, we might want maximum removal of dirt in minimum time. Being more sophisticated, we also don't want the car to get damaged, or to use too much water or energy...

And we might want the robot to have spare time at the weekend to write its novel...

So: the choice of a performance measure is itself worthy of careful consideration.

Measuring performance

We're usually interested in *expected, long-term performance*.

We are generally interested in *expected* performance because usually agents are not *omniscient*—they don't *infallibly* know the outcome of their actions.

It is *rational* for you to enter this lecture theatre even if the roof falls in today.

An agent capable of detecting and protecting itself from a falling roof might be more *successful* than you, but *not* more *rational*.

Measuring performance

We generally also favour *long-term performance*.

For example: we'd probably prefer a spam filter that detects spam most of the time over a long time period, over one that has 100 percent accuracy on the first day and dismal performance thereafter.

Environments

How can an agent's *environment* affect its design? *Example:* the environment for a *chess program* is vastly different to that for an *autonomous deep-space vehicle*. Some common attributes of an environment have a considerable influence on agent design.

- *Accessible/inaccessible:* do percepts tell you *everything* you need to know about the world?
- *Deterministic/non-deterministic:* does the future depend *predictably* on the present and your actions?
- *Episodic/non-episodic* is the agent run in independent episodes.
- *Static/dynamic:* can the world change while the agent is deciding what to do?
- *Discrete/continuous:* an environment is discrete if the sets of allowable percepts and actions are finite.

Environments

All of this assumes there is only one agent.

When multiple agents are involved we need to consider:

- Whether the situation is *competitive* or *cooperative*.
- Whether *communication* required?

An example of multiple agents:

news.bbc.co.uk/1/hi/technology/3486335.stm

Basic structures for intelligent agents

Are there sensible ways in which to think about the *structure* of an agent? Again, this is likely to be *problem-specific*, although perhaps to a lesser extent.

So far, an agent is based on percepts, actions and goals.

Example: automatic aircraft pilot.

Percepts: sensor information regarding height, speed, engines *etc*, audio and video inputs, and so on.

Actions: manipulation of the aircraft's controls.

Also, perhaps talking to the passengers *etc*.

Goals: get to the necessary destination as quickly as possible with minimal use of fuel, without crashing *etc*.

Programming agents

A basic agent might be thought of as follows:

```
action agent(percept)
{
   static memory;      // the agent's memory.

   memory = updateMemory(memory,percept);
   nextAction = chooseAction(memory);
   memory = updateMemory(memory,nextAction);

   return nextAction;
}
```

Obviously a great deal of complexity has been hidden inside `updateMemory` and `chooseAction`.

Also, further complexity has been ignored. For example, what if a percept arrives while `chooseAction` is executing?

Programming agents

We'll initially look at two hopelessly limited approaches, because they do suggest a couple of important points.

Hopelessly limited approach number 1: use a table to map percept sequences to actions. This can quickly be rejected.

- The table will be *huge* for any problem of interest. About 35^{100} entries for a chess player.
- We don't usually know how to fill the table.
- Even if we allow table entries to be *learned* it will take too long.
- The system would have no *autonomy*.

We can attempt to overcome these problems by allowing agents to *reason*.

Autonomy is an interesting issue though...

Autonomy

If an agent's behaviour depends in some manner on its *own experience of the world* via its percept sequence, we say it is *autonomous*.

- An agent using only built-in knowledge would seem not to be successful at AI in any meaningful sense: its behaviour is predefined by its designer.
- On the other hand *some* built-in knowledge seems essential, even to humans.

Not all animals are entirely autonomous.

For example: dung beetles.

Reflex agents

Hopelessly limited approach number 2: try *extracting* pertinent information and using *rules* based on this.

Condition-action rules:

`if` a certain *state* is observed `then` perform some *action*

Example:

`if` speed has fallen below that required `then` increase power to the engines

This leads to a *reflex agent*.

Keeping track of the environment

Some points immediately present themselves regarding *why* reflex agents are unsatisfactory:

- We can't always decide what to do based on the *current percept*.
- However storing *all* past percepts might be undesirable (for example requiring too much memory) or just unnecessary.
- Reflex agents don't maintain a description of the *state of their environment*...
- ...however this seems necessary for any meaningful AI. (Consider automating the task of driving.)

This is all the more important as usually percepts don't tell you *everything about the state*.

Keeping track of the environment

It seems reasonable that an agent should maintain:

- A *description of the current state of its environment*.
- Knowledge of how the environment *changes independently of the agent*.
- Knowledge of how the agent's *actions affect its environment*.

This requires us to do *knowledge representation* and *reasoning*.

Goal-based agents

It seems reasonable that an agent should choose a rational course of action depending on its *goal*.

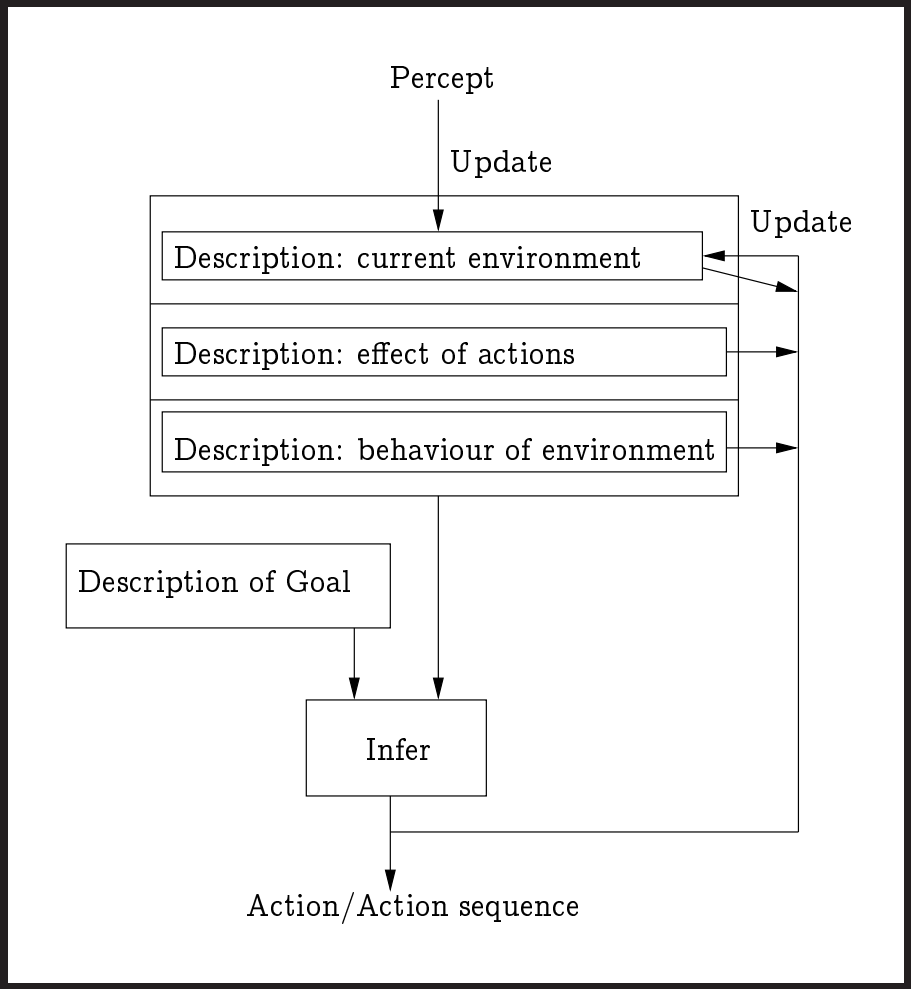
- If an agent has knowledge of how its actions affect the environment, then it has a basis for choosing actions to achieve goals.
- To obtain a *sequence* of actions we need to be able to *search* and to *plan*.

This is *fundamentally different* from a reflex agent.

For example: by changing the goal you can change the entire behaviour.

Goal-based agents

We now have a basic design that looks something like this:



Utility-based agents

Introducing goals is still not the end of the story.

There may be *many* sequences of actions that lead to a given goal, and *some may be preferable to others*.

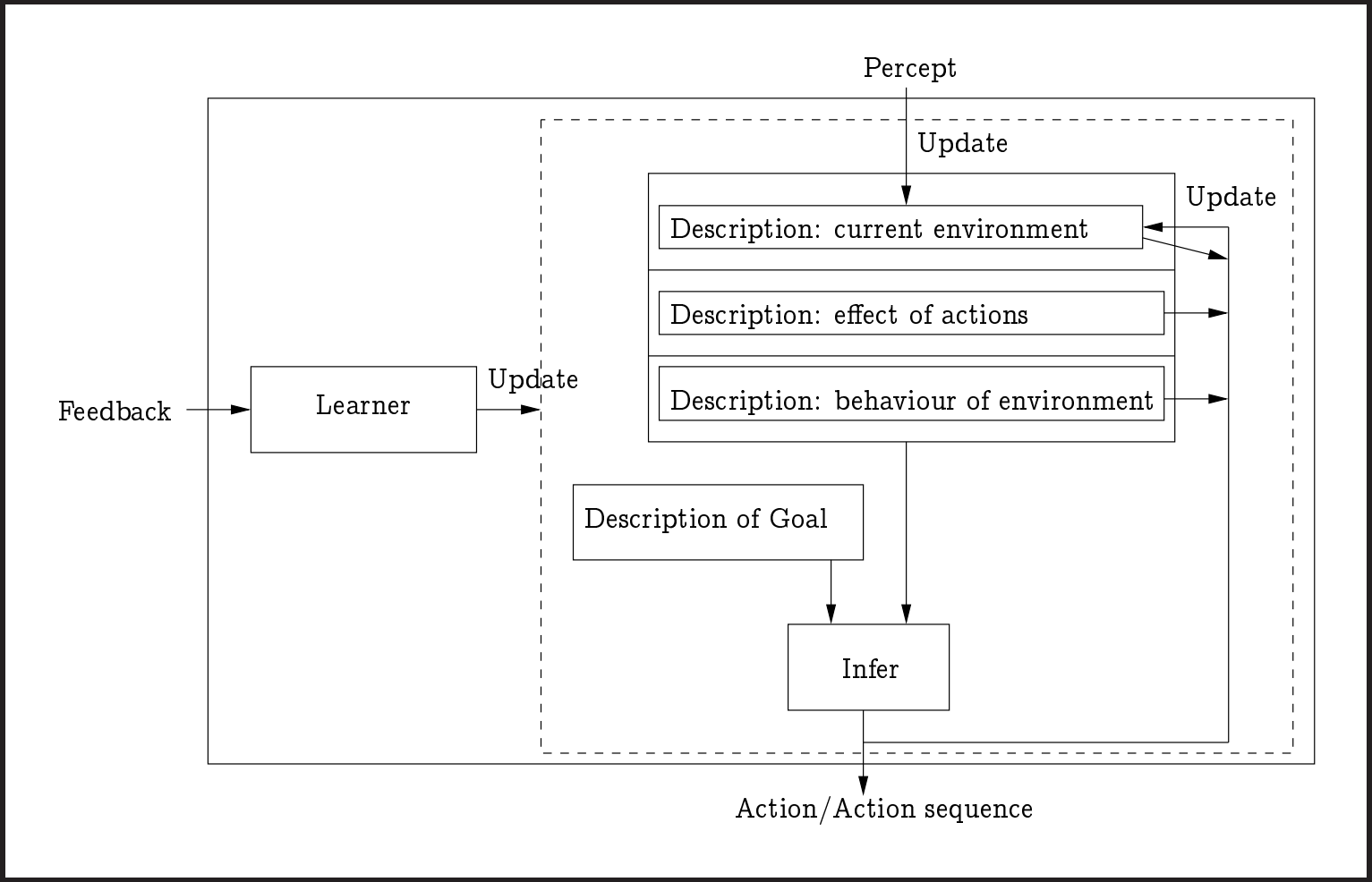
A *utility function* maps a state to a number representing the desirability of that state.

- We can trade-off *conflicting goals*, for example speed and safety.
- If an agent has several goals and is not certain of achieving any of them, then it can trade-off likelihood of reaching a goal against the desirability of getting there.

Maximising expected utility over time forms a fundamental model for the design of agents. However we don't get as far as that until AI II.

Learning agents

It seems reasonable that an agent should *learn from experience*.



Learning agents

This requires two additions:

- The learner needs some form of *feedback* on the agent's performance. This can come in several different forms.
- In general, we also need a means of *generating new behaviour* in order to find out about the world.

This in turn implies a trade-off: should the agent spend time *exploiting* what it's learned so far, or *exploring* the environment on the basis that it might learn something really useful?

What have we learned? (No pun intended...)

The *crucial* things that should be taken away from this lecture are:

- The nature of an agent depends on its *environment* and *performance measure*.
- We're usually interested in *expected, long-term performance*.
- Autonomy requires that an agent in some way behaves *depending on its experience of the world*.
- There is a *natural basic structure* on which agent design can be based.
- Consideration of that structure leads naturally to the basic areas covered in this course.

Those basic areas are: *knowledge representation and reasoning, search, planning and learning*. Oh, and finally, we've learned NOT TO MESS WITH EVIL ROBOT... he's a VERY BAD ROBOT!

Exercise

It is notoriously difficult to predict what will be possible in the future, so your answers might well be amusing to you when you find them in twenty years time.

Exercise:

1. If you haven't seen it already, watch the film *A.I. Artificial Intelligence* paying particular attention to the character *"Teddy"*.
2. A large number of subjects were covered in the initial lectures in terms of how they've influenced AI: for example philosophy, mathematics, economics and so on. How do these show up in Teddy's design?
3. What aspects of Teddy are within our current capabilities to design?
4. What aspects of Teddy would you expect to be able to implement within the next fifteen years. How about the next fifty years?
5. Are there aspects of Teddy that you would expect to elude us for one hundred years or more?
6. To what extent does the "natural basic structure" for an agent, as described in these notes, form a useful basis for implementing Teddy's internals? What is missing?