4

Practical Reasoning Agents

Whatever the merits of agents that decide what to do by proving theorems, it seems clear that *we* do not use purely logical reasoning in order to decide what to do. Certainly something like logical reasoning can play a part, but a moment's reflection should confirm that for most of the time, very different processes are taking place. In this chapter, I will focus on a model of agency that takes its inspiration from the processes that seem to take place as we decide what to do.

4.1 Practical Reasoning Equals Deliberation Plus Means-Ends Reasoning

The particular model of decision making is known as *practical reasoning*. Practical reasoning is reasoning directed towards actions - the process of figuring out what to do.

> Practical reasoning is a matter of weighing conflicting considerations for and against competing options, where the relevant considerations are provided by what the agent desires/values/cares about and what the agent believes.
>
> (Bratman, 1990, p. 17)

It is important to distinguish practical reasoning from *theoretical reasoning* (Eliasmith, 1999). Theoretical reasoning is directed towards beliefs. To use a rather tired example, if I believe that all men are mortal, and I believe that Socrates is a man, then I will usually conclude that Socrates is mortal. The process of concluding that Socrates is mortal is theoretical reasoning, since it affects only my beliefs about the world. The process of deciding to catch a bus instead of a train, however, is practical reasoning, since it is reasoning directed towards action.

Human practical reasoning appears to consist of at least two distinct activities. The first of these involves deciding *what* state of affairs we want to achieve; the second process involves deciding *how* we want to achieve these states of affairs. The former process - deciding what states of affairs to achieve - is known as *deliberation*. The latter process - deciding how to achieve these states of affairs - we call *means-ends reasoning*.

To better understand deliberation and means-ends reasoning, consider the following example. When a person graduates from university with a first degree, he or she is faced with some important choices. Typically, one proceeds in these choices by first deciding what sort of career to follow. For example, one might consider a career as an academic, or a career in industry. The process of deciding which career to aim for is deliberation. Once one has fixed upon a career, there are further choices to be made; in particular, how to bring about this career. Suppose that after deliberation, you choose to pursue a career as an academic. The next step is to decide *how to achieve* this state of affairs. This process is means-ends reasoning. The end result of means-ends reasoning is a *plan* or *recipe* of some kind for achieving the chosen state of affairs. For the career example, a plan might involve first applying to an appropriate university for a PhD place, and so on. After obtaining a plan, an agent will typically then attempt to carry out (or *execute*) the plan, in order to bring about the chosen state of affairs. If all goes well (the plan is sound, and the agent's environment cooperates sufficiently), then after the plan has been executed, the chosen state of affairs will be achieved.

Thus described, practical reasoning seems a straightforward process, and in an ideal world, it would be. But there are several complications. The first is that deliberation and means-ends reasoning are *computational* processes. In all real agents (and, in particular, artificial agents), such computational processes will take place under *resource bounds*. By this I mean that an agent will only have a fixed amount of memory and a fixed processor available to carry out its computations. Together, these resource bounds impose a limit on the size of computations that can be carried out in any given amount of time. No real agent will be able to carry out arbitrarily large computations in a finite amount of time. Since almost any real environment will also operate in the presence of *time constraints* of some kind, this means that means-ends reasoning and deliberation must be carried out in a fixed, finite number of processor cycles, with a fixed, finite amount of memory space. From this discussion, we can see that resource bounds have two important implications:

- Computation is a valuable resource for agents situated in real-time environments. The ability to perform well will be determined at least in part by the ability to make efficient use of available computational resources. In other words, an agent must *control* its reasoning effectively if it is to perform well.
- Agents cannot deliberate indefinitely. They must clearly *stop* deliberating at some point, having chosen some state of affairs, and commit to achieving this state of affairs. It may well be that the state of affairs it has fixed upon is not optimal – further deliberation may have led it to fix upon an another state of affairs.

We refer to the states of affairs that an agent has chosen and committed to as its *intentions.*

Intentions in practical reasoning

First, notice that it is possible to distinguish several different types of intention. In ordinary speech, we use the term 'intention' to characterize both *actions* and *states of mind.* To adapt an example from Bratman (Bratman, 1987, p. 1), I might intentionally push someone under a train, and push them with the intention of killing them. Intention is here used to characterize an action – the action of pushing someone under a train. Alternatively, I might have the intention this morning of pushing someone under a train this afternoon. Here, intention is used to characterize my state of mind. In this book, when I talk about intentions, I mean intentions as states of mind. In particular, I mean *future-directed intentions* – intentions that an agent has towards some future state of affairs.

The most obvious role of intentions is that they are *pro-attitudes* (Bratman, 1990, p. 23). By this, I mean that they tend to lead to action. Suppose I have an intention to write a book. If I truly have such an intention, then you would expect me to make a *reasonable attempt* to achieve it. This would usually involve, at the very least, me initiating some plan of action that I believed would satisfy the intention. In this sense, intentions tend to play a primary role in the production of action. As time passes, and my intention about the future becomes my intention about the present, then it plays a direct role in the production of action. Of course, having an intention does not necessarily lead to action. For example, I can have an intention now to attend a conference later in the year. I can be utterly sincere in this intention, and yet if I learn of some event that must take precedence over the conference, I may never even get as far as considering travel arrangements.

Bratman notes that intentions play a much stronger role in influencing action than other pro-attitudes, such as mere desires.

> My desire to play basketball this afternoon is merely a potential influencer of my conduct this afternoon. It must vie with my other relevant desires...before it is settled what I will do. In contrast, once I intend

> to play basketball this afternoon, the matter is settled: I normally need not continue to weigh the pros and cons. When the afternoon arrives, I will normally just proceed to execute my intentions.
>
> (Bratman, 1990, p. 22)

The second main property of intentions is that they *persist*. If I adopt an intention to become an academic, then I should persist with this intention and attempt to achieve it. For if I immediately drop my intentions without devoting any resources to achieving them, then I will not be acting rationally. Indeed, you might be inclined to say that I never really had intentions in the first place.

Of course, I should not persist with my intention for too long – if it becomes clear to me that I will never become an academic, then it is only rational to drop my intention to do so. Similarly, if the reason for having an intention goes away, then it would be rational for me to drop the intention. For example, if I adopted the intention to become an academic because I believed it would be an easy life, but then discover that this is not the case (e.g. I might be expected to actually teach!), then the justification for the intention is no longer present, and I should drop the intention.

If I initially fail to achieve an intention, then you would expect me to *try again* – you would not expect me to simply give up. For example, if my first application for a PhD program is rejected, then you might expect me to apply to alternative universities.

The third main property of intentions is that once I have adopted an intention, the very fact of having this intention will constrain my future practical reasoning. For example, while I hold some particular intention, I will not subsequently entertain options that are *inconsistent* with that intention. Intending to write a book, for example, would preclude the option of partying every night: the two are mutually exclusive. This is in fact a highly desirable property from the point of view of implementing rational agents, because in providing a 'filter of admissibility', intentions can be seen to constrain the space of possible intentions that an agent needs to consider.

Finally, intentions are closely related to beliefs about the future. For example, if I intend to become an academic, then I should believe that, assuming some certain background conditions are satisfied, I will indeed become an academic. For if I truly believe that I will never be an academic, it would be nonsensical of me to have an intention to become one. Thus if I intend to become an academic, I should at least believe that there is a good chance I will indeed become one. However, there is what appears at first sight to be a paradox here. While I might believe that I will indeed succeed in achieving my intention, if I am rational, then I must also recognize the possibility that I can *fail* to bring it about – that there is some circumstance under which my intention is not satisfied.

From this discussion, we can identify the following closely related situations.

- Having an intention to bring about φ, while believing that you will not bring about φ is called *intention-belief inconsistency*, and is not rational (see, for example, Bratman, 1987, pp. 37, 38).
- Having an intention to achieve φ without believing that φ will be the case is *intention-belief incompleteness*, and is an acceptable property of rational agents (see, for example, Bratman, 1987, p. 38).

The distinction between these two cases is known as the *asymmetry thesis* (Bratman, 1987, pp. 37–41).

Summarizing, we can see that intentions play the following important roles in practical reasoning.

Intentions drive means-ends reasoning. If I have formed an intention, then I will attempt to achieve the intention, which involves, among other things, deciding *how* to achieve it. Moreover, if one particular course of action fails to achieve an intention, then I will typically attempt others.

Intentions persist. I will not usually give up on my intentions without good reason – they will persist, typically until I believe I have successfully achieved them, I believe I cannot achieve them, or I believe the reason for the intention is no longer present.

Intentions constrain future deliberation. I will not entertain options that are inconsistent with my current intentions.

Intentions influence beliefs upon which future practical reasoning is based. If I adopt an intention, then I can plan for the future on the assumption that I will achieve the intention. For if I intend to achieve some state of affairs while simultaneously believing that I will not achieve it, then I am being irrational.

Notice from this discussion that intentions *interact* with an agent's beliefs and other mental states. For example, having an intention to φ implies that I do not believe φ is impossible, and moreover that I believe given the right circumstances, φ will be achieved. However, satisfactorily capturing the interaction between intention and belief turns out to be surprisingly hard – some discussion on this topic appears in Chapter 12.

Throughout the remainder of this chapter, I make one important assumption: that the agent maintains some explicit *representation* of its beliefs, desires, and intentions. However, I will not be concerned with *how* beliefs and the like are represented. One possibility is that they are represented *symbolically*, for example as logical statements *a là* Prolog facts (Clocksin and Mellish, 1981). However, the assumption that beliefs, desires, and intentions are symbolically represented is by no means necessary for the remainder of the book. I use B to denote a variable that holds the agent's current beliefs, and let Bel be the set of all such beliefs. Similarly, I use D as a variable for desires, and Des to denote the set of all desires.

Finally, the variable I represents the agent's intentions, and Int is the set of all possible intentions.

In what follows, deliberation will be modelled via two functions:

- an option generation function; and
- a filtering function.

The signature of the option generation function $options$ is as follows:

$$options : \wp(Bel) \times \wp(Int) \rightarrow \wp(Des).$$

This function takes the agent's current beliefs and current intentions, and on the basis of these produces a set of possible options or desires.

In order to select between competing options, an agent uses a *filter* function. Intuitively, the filter function must simply select the 'best' option(s) for the agent to commit to. We represent the filter process through a function $filter$, with a signature as follows:

$$filter : \wp(Bel) \times \wp(Des) \times \wp(Int) \rightarrow \wp(Int).$$

An agent's belief update process is modelled through a *belief revision function*:

$$brf : \wp(Bel) \times Per \rightarrow \wp(Bel).$$

4.2 Means–Ends Reasoning

Means-ends reasoning is the process of deciding how to achieve an end (i.e. an intention that you have) using the available means (i.e. the actions that you can perform). Means-ends reasoning is perhaps better known in the AI community as *planning*.

Planning is essentially automatic programming. A planner is a system that takes as input representations of the following.

(1) A *goal*, *intention* or (in the terminology of Chapter 2) a *task*. This is something that the agent wants to achieve (in the case of achievement tasks - see Chapter 2), or a state of affairs that the agent wants to maintain or avoid (in the case of maintenance tasks - see Chapter 2).

(2) The current *state of the environment* - the agent's *beliefs*.

(3) The *actions* available to the agent.

As output, a planning algorithm generates a *plan* (see Figure 4.1). This is a course of action - a 'recipe'. If the planning algorithm does its job correctly, then if the agent executes this plan ('follows the recipe') from a state in which the world is as described in (2), then once the plan has been completely executed, the goal/intention/task described in (1) will be carried out.

The first real planner was the STRIPS system, developed by Fikes in the late 1960s/early 1970s (Fikes and Nilsson, 1971). The two basic components of STRIPS

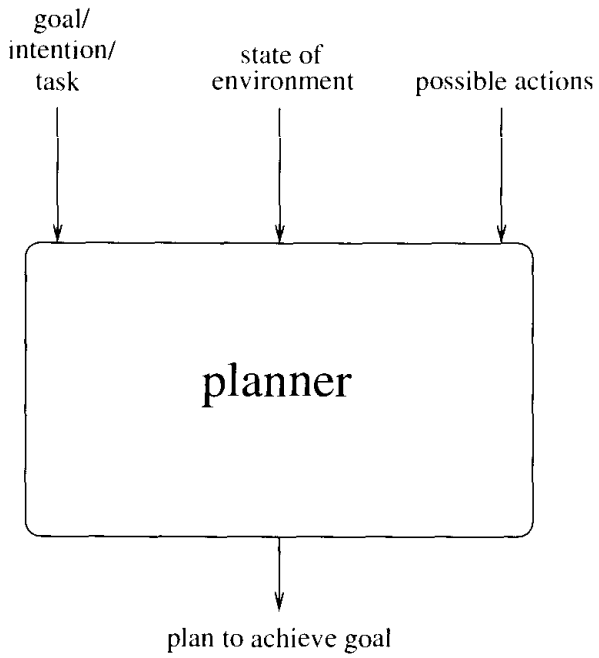

Figure 4.1 Planning.

were a model of the world as a set of formulae of first-order logic, and a set of *action schemata*, which describe the preconditions and effects of all the actions available to the planning agent. This latter component has perhaps proved to be STRIPS' most lasting legacy in the AI planning community: nearly all implemented planners employ the 'STRIPS formalism' for action, or some variant of it. The STRIPS planning algorithm was based on a principle of finding the 'difference' between the current state of the world and the goal state, and reducing this difference by applying an action. Unfortunately, this proved to be an inefficient process for formulating plans, as STRIPS tended to become 'lost' in low-level plan detail.

There is not scope in this book to give a detailed technical introduction to planning algorithms and technologies, and in fact it is probably not appropriate to do so. Nevertheless, it is at least worth giving a short overview of the main concepts.

The Blocks World

In time-honoured fashion, I will illustrate the techniques with reference to a *Blocks World*. The Blocks World contains three blocks (A, B, and C) of equal size, a robot arm capable of picking up and moving one block at a time, and a table top. The blocks may be placed on the table top, or one may be placed on top of the other. Figure 4.2 shows one possible configuration of the Blocks World.

Notice that in the description of planning algorithms I gave above, I stated that planning algorithms take as input *representations* of the goal, the current state of the environment, and the actions available. The first issue is exactly what form these representations take. The STRIPS system made use of representations based

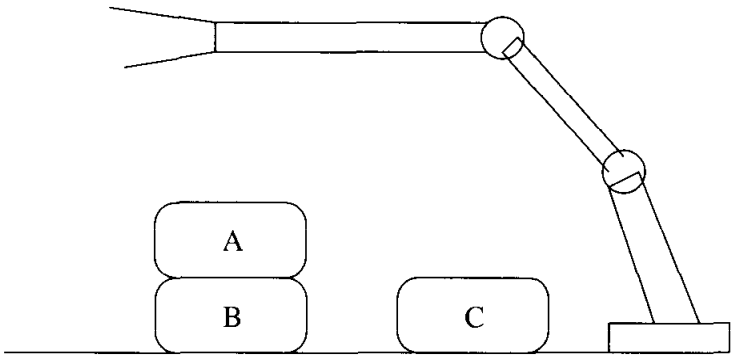

Figure 4.2 The Blocks World.

on first-order logic. I will use the predicates in Table 4.1 to represent the Blocks World.

A description of the Blocks World in Figure 4.2 is using these predicates as follows:

$$\{Clear(A), On(A, B), OnTable(B), OnTable(C), Clear(C)\}.$$

I am implicitly making use of the *closed world* assumption: if something is not explicitly stated to be true, then it is assumed false.

The next issue is how to represent goals. Again, we represent a goal as a set of formulae of first-order logic:

$$\{OnTable(A), OnTable(B), OnTable(C)\}.$$

So the goal is that all the blocks are on the table. To represent actions, we make use of the precondition/delete/add list notation – the STRIPS formalism. In this formalism, each action has

- a *name* – which may have arguments;
- a *precondition list* – a list of facts which must be true for the action to be executed;
- a *delete list* – a list of facts that are no longer true after the action is performed; and
- an *add list* – a list of facts made true by executing the action.

The *stack* action occurs when the robot arm places the object x it is holding on top of object y:

$$\begin{array}{ll} Stack(x, y) & \\ \text{pre} & \{Clear(y), Holding(x)\} \\ \text{del} & \{Clear(y), Holding(x)\} \\ \text{add} & \{ArmEmpty y, On(x, y)\} \end{array}$$

The *unstack* action occurs when the robot arm picks an object x up from on top of another object y:

Table 4.1 Predicates for describing the Blocks World.

Predicate	Meaning
$On(x, y)$	object x on top of object y
$OnTable(x)$	object x is on the table
$Clear(x)$	nothing is on top of object x
$Holding(x)$	robot arm is holding x
$ArmEmpty$	robot arm empty (not holding anything)

$UnStack(x, y)$
pre $\{On(x, y), Clear(x), ArmEmpty\}$
del $\{On(x, y), ArmEmpty\}$
add $\{Holding(x), Clear(y)\}$

The *pickup* action occurs when the arm picks up an object x from the table:

$Pickup(x)$
pre $\{Clear(x), OnTable(x), ArmEmpty\}$
del $\{OnTable(x), ArmEmpty\}$
add $\{Holding(x)\}$

The *putdown* action occurs when the arm places the object x onto the table:

$PutDown(x)$
pre $\{Holding(x)\}$
del $\{Holding(x)\}$
add $\{ArmEmpty, OnTable(x)\}$

Let us now describe what is going on somewhat more formally. First, as we have throughout the book, we assume a fixed set of actions $Ac = \{\alpha_1, \ldots, \alpha_n\}$ that the agent can perform. A *descriptor* for an action $\alpha \in Ac$ is a triple

$$\langle P_\alpha, D_\alpha, A_\alpha \rangle,$$

where

- P_α is a set of formulae of first-order logic that characterize the *precondition* of action α;
- D_α is a set of formulae of first-order logic that characterize those facts made *false* by the performance of α (the *delete list*); and
- A_α is a set of formulae of first-order logic that characterize those facts made *true* by the performance of α (the *add list*).

For simplicity, we will assume that the precondition, delete, and add lists are constrained to only contain *ground atoms* - individual predicates, which do not contain logical connectives or variables.

A *planning problem* (over the set of actions Ac) is then determined by a triple

$$\langle \Delta, O, \gamma \rangle,$$

where

- Δ is the beliefs of the agent about the *initial* state of the world - these beliefs will be a set of formulae of first order (cf. the vacuum world in Chapter 2);
- $O = \{\langle P_\alpha, D_\alpha, A_\alpha \rangle \mid \alpha \in Ac\}$ is an indexed set of operator descriptors, one for each available action α; and
- γ is a set of formulae of first-order logic, representing the *goal/task/intention* to be achieved.

A *plan* π is a sequence of actions

$$\pi = (\alpha_1, \ldots, \alpha_n),$$

where each α_i is a member of Ac.

With respect to a planning problem $\langle \Delta, O, \gamma \rangle$, a plan $\pi = (\alpha_1, \ldots, \alpha_n)$ determines a sequence of $n + 1$ *environment models*

$$\Delta_0, \Delta_1, \ldots, \Delta_n,$$

where

$$\Delta_0 = \Delta$$

and

$$\Delta_i = (\Delta_{i-1} \setminus D_{\alpha_i}) \cup A_{\alpha_i} \quad \text{for } 1 \leqslant i \leqslant n.$$

A (linear) plan $\pi = (\alpha_1, \ldots, \alpha_n)$ is said to be *acceptable* with respect to the problem $\langle \Delta, O, \gamma \rangle$ if, and only if, the precondition of every action is satisfied in the preceding environment model, i.e. if $\Delta_{i-1} \vDash P_{\alpha_i}$, for all $1 \leqslant i \leqslant n$. A plan $\pi = (\alpha_1, \ldots, \alpha_n)$ is *correct* with respect to $\langle \Delta, O, \gamma \rangle$ if and only if

(1) it is acceptable; and

(2) $\Delta_n \vDash \gamma$ (i.e. if the goal is achieved in the final environment state generated by the plan).

The problem to be solved by a planning system can then be stated as follows.

> Given a planning problem $\langle \Delta, O, \gamma \rangle$, find a correct plan for $\langle \Delta, O, \gamma \rangle$ or else announce that none exists).

(It is worth comparing this discussion with that on the synthesis of agents in Chapter 3 - similar comments apply with respect to the issues of soundness and completeness.)

We will use π (with decorations: $\pi', \pi_1, \ldots$) to denote plans, and let $Plan$ be the set of all plans (over some set of actions Ac). We will make use of a number of auxiliary definitions for manipulating plans (some of these will not actually be required until later in this chapter):

- if π is a plan, then we write $pre(\pi)$ to denote the precondition of π, and $body(\pi)$ to denote the body of π;
- if π is a plan, then we write $empty(\pi)$ to mean that plan π is the empty sequence (thus $empty(\ldots)$ is a Boolean-valued function);
- $execute(\ldots)$ is a procedure that takes as input a single plan and executes it without stopping - executing a plan simply means executing each action in the plan body in turn;
- if π is a plan, then by $head(\pi)$ we mean the plan made up of the first action in the plan body of π; for example, if the body of π is $\alpha_1, \ldots, \alpha_n$, then the body of $head(\pi)$ contains only the action α_1;
- if π is a plan, then by $tail(\pi)$ we mean the plan made up of all but the first action in the plan body of π; for example, if the body of π is $\alpha_1, \alpha_2, \ldots, \alpha_n$, then the body of $tail(\pi)$ contains actions $\alpha_2, \ldots, \alpha_n$;
- if π is a plan, $I \subseteq Int$ is a set of intentions, and $B \subseteq Bel$ is a set of beliefs, then we write $sound(\pi, I, B)$ to mean that π is a correct plan for intentions I given beliefs B (Lifschitz, 1986).

An agent's means–ends reasoning capability is represented by a function

$$plan : \wp(Bel) \times \wp(Int) \times \wp(Ac) \rightarrow Plan,$$

which, on the basis of an agent's current beliefs and current intentions, determines a plan to achieve the intentions.

Notice that there is nothing in the definition of the $plan(\ldots)$ function which requires an agent to engage in *plan generation* - constructing a plan from scratch (Allen *et al.*, 1990). In many implemented practical reasoning agents, the $plan(\ldots)$ function is implemented by giving the agent a *plan library* (Georgeff and Lansky, 1987). A plan library is a pre-assembled collection of plans, which an agent designer gives to an agent. Finding a plan to achieve an intention then simply involves a single pass through the plan library to find a plan that, when executed, will have the intention as a postcondition, and will be sound given the agent's current beliefs. Preconditions and postconditions for plans are often represented as (lists of) atoms of first-order logic, and beliefs and intentions as ground atoms of first-order logic. Finding a plan to achieve an intention then reduces to finding a plan whose precondition unifies with the agent's beliefs, and whose postcondition unifies with the intention. At the end of this chapter, we will see how this idea works in the PRS system.

4.3 Implementing a Practical Reasoning Agent

We can now discuss the overall control structure of a practical reasoning agent. Figure 4.3 gives the pseudo-code for the control cycle of such an agent. The basic structure of the decision-making process is a loop, in which the agent continually

```
Algorithm: Practical Reasoning Agent Control Loop
1.
2.  B ← B0;        /* B0 are initial beliefs */
3.  I ← I0;        /* I0 are initial intentions */
4.  while true do
5.      get next percept ρ through see(...) function;
6.      B ← brf(B,ρ);
7.      D ← options(B,I);
8.      I ← filter(B,D,I);
9.      π ← plan(B,I,Ac);
10.     while not (empty(π) or succeeded(I,B) or impossible(I,B)) do
11.         α ← head(π);
12.         execute(α);
13.         π ← tail(π);
14.         get next percept ρ through see(...) function;
15.         B ← brf(B,ρ);
16.         if reconsider(I,B) then
17.             D ← options(B,I);
18.             I ← filter(B,D,I);
19.         end-if
20.         if not sound(π,I,B) then
21.             π ← plan(B,I,Ac)
22.         end-if
23.     end-while
24. end-while
```

Figure 4.3 A practical reasoning agent.

- observes the world, and updates beliefs;
- deliberates to decide what intention to achieve (deliberation being done by first determining the available options and then by filtering);
- uses means-ends reasoning to find a plan to achieve these intentions;
- executes the plan.

However, this basic control loop is complicated by a number of concerns. The first of these is that of *commitment* - and, in particular, how committed an agent is to both ends (the intention) and means (the plan to achieve the intention).

Commitment to ends and means

When an option successfully passes through the $filter$ function and is hence chosen by the agent as an intention, we say that the agent has made a *commitment* to that option. Commitment implies *temporal persistence* - an intention, once adopted, should not immediately evaporate. A critical issue is just *how* committed an agent should be to its intentions. That is, how long should an intention persist? Under what circumstances should an intention vanish?

To motivate the discussion further, consider the following scenario.

> Some time in the not-so-distant future, you are having trouble with your new household robot. You say "Willie, bring me a beer." The robot replies "OK boss." Twenty minutes later, you screech "Willie, why didn't you bring me that beer?" It answers "Well, I intended to get you the beer, but I decided to do something else." Miffed, you send the wise guy back to the manufacturer, complaining about a lack of commitment. After retrofitting, Willie is returned, marked "Model C: The Committed Assistant." Again, you ask Willie to bring you a beer. Again, it accedes, replying "Sure thing." Then you ask: "What kind of beer did you buy?" It answers: "Genessee." You say "Never mind." One minute later, Willie trundles over with a Genessee in its gripper. This time, you angrily return Willie for overcommitment. After still more tinkering, the manufacturer sends Willie back, promising no more problems with its commitments. So, being a somewhat trusting customer, you accept the rascal back into your household, but as a test, you ask it to bring you your last beer. Willie again accedes, saying "Yes, Sir." (Its attitude problem seems to have been fixed.) The robot gets the beer and starts towards you. As it approaches, it lifts its arm, wheels around, deliberately smashes the bottle, and trundles off. Back at the plant, when interrogated by customer service as to why it had abandoned its commitments, the robot replies that according to its specifications, it kept its commitments as long as required - commitments must be dropped when fulfilled or impossible to achieve. By smashing the bottle, the commitment became unachievable.
>
> (Cohen and Levesque, 1990a, pp. 213, 214)

The mechanism an agent uses to determine when and how to drop intentions is known as a *commitment strategy*. The following three commitment strategies are commonly discussed in the literature of rational agents (Rao and Georgeff, 1991b).

Blind commitment. A blindly committed agent will continue to maintain an intention until it believes the intention has actually been achieved. Blind commitment is also sometimes referred to as *fanatical* commitment.

Single-minded commitment. A single-minded agent will continue to maintain an intention until it believes that either the intention has been achieved, or else that it is no longer possible to achieve the intention.

Open-minded commitment. An open-minded agent will maintain an intention as long as it is still believed possible.

Note that an agent has commitment both to *ends* (i.e. the state of affairs it wishes to bring about) and *means* (i.e. the mechanism via which the agent wishes to achieve the state of affairs).

With respect to commitment to means (i.e. plans), the solution adopted in Figure 4.3 is as follows. An agent will maintain a commitment to an intention until (i) it believes the intention has succeeded; (ii) it believes the intention is impossible, or (iii) there is nothing left to execute in the plan. This is single-minded commitment. I write $succeeded(I, B)$ to mean that given beliefs B, the intentions I can be regarded as having been satisfied. Similarly, we write $impossible(I, B)$ to mean that intentions I are impossible given beliefs B. The main loop, capturing this commitment to means, is in lines (10)–(23).

How about commitment to ends? When should an agent stop to *reconsider* its intentions? One possibility is to reconsider intentions at every opportunity – in particular, after executing every possible action. If option generation and filtering were computationally cheap processes, then this would be an acceptable strategy. Unfortunately, we know that deliberation is not cheap – it takes a considerable amount of time. While the agent is deliberating, the environment in which the agent is working is changing, possibly rendering its newly formed intentions irrelevant.

We are thus presented with a dilemma:

- an agent that does not stop to reconsider its intentions sufficiently often will continue attempting to achieve its intentions even after it is clear that they cannot be achieved, or that there is no longer any reason for achieving them;
- an agent that *constantly* reconsiders its attentions may spend insufficient time actually working to achieve them, and hence runs the risk of never actually achieving them.

There is clearly a trade-off to be struck between the degree of commitment and reconsideration at work here. To try to capture this trade-off, Figure 4.3 incorporates an explicit *meta-level control* component. The idea is to have a Boolean-valued function, $reconsider$, such that $reconsider(I, B)$ evaluates to 'true' just in case it is appropriate for the agent with beliefs B and intentions I to reconsider its intentions. Deciding whether to reconsider intentions thus falls to this function.

It is interesting to consider the circumstances under which this function can be said to behave *optimally*. Suppose that the agent's deliberation and plan generation functions are in some sense perfect: that deliberation always chooses the 'best' intentions (however that is defined for the application at hand), and planning always produces an appropriate plan. Further suppose that time expended always has a cost – the agent does not benefit by doing nothing. Then it is not difficult to see that the function $reconsider(\ldots)$ will be behaving optimally if, and only if, whenever it chooses to deliberate, the agent changes intentions (Wooldridge and Parsons, 1999). For if the agent chose to deliberate but did not change intentions, then the effort expended on deliberation was wasted. Similarly, if an agent

Table 4.2 Practical reasoning situations (cf. Bratman *et al.*, 1988, p. 353).

Situation number	Chose to deliberate?	Changed intentions?	Would have changed intentions?	*reconsider*(...) optimal?
1.	No	—	No	Yes
2.	No	—	Yes	No
3.	Yes	No	—	No
4.	Yes	Yes	—	Yes

should have changed intentions, but failed to do so, then the effort expended on attempting to achieve its intentions was also wasted.

The possible interactions between deliberation and meta-level control (the function *reconsider*(...)) are summarized in Table 4.2.

- In situation (1), the agent did not choose to deliberate, and as a consequence, did not choose to change intentions. Moreover, if it *had* chosen to deliberate, it would not have changed intentions. In this situation, the *reconsider*(...) function is behaving optimally.
- In situation (2), the agent did not choose to deliberate, but if it had done so, it *would* have changed intentions. In this situation, the *reconsider*(...) function is not behaving optimally.
- In situation (3), the agent chose to deliberate, but did not change intentions. In this situation, the *reconsider*(...) function is not behaving optimally.
- In situation (4), the agent chose to deliberate, and did change intentions. In this situation, the *reconsider*(...) function is behaving optimally.

Notice that there is an important assumption implicit within this discussion: that the cost of executing the *reconsider*(...) function is *much* less than the cost of the deliberation process itself. Otherwise, the *reconsider*(...) function could simply use the deliberation process as an oracle, running it as a subroutine and choosing to deliberate just in case the deliberation process changed intentions.

The nature of the trade-off was examined by David Kinny and Michael Georgeff in a number of experiments carried out using a BDI agent system (Kinny and Georgeff, 1991). The aims of Kinny and Georgeff's investigation were to

> (1) assess the feasibility of experimentally measuring agent effectiveness in a simulated environment (2) investigate how commitment to goals contributes to effective agent behaviour and (3) compare the properties of different strategies for reacting to change.
>
> (Kinny and Georgeff, 1991, p. 82)

In Kinny and Georgeff's experiments, two different types of reconsideration strategy were used: *bold* agents, which never pause to reconsider their intentions

before their current plan is fully executed; and *cautious* agents, which stop to reconsider after the execution of every action. These characteristics are defined by a *degree of boldness*, which specifies the maximum number of plan steps the agent executes before reconsidering its intentions. Dynamism in the environment is represented by the *rate of environment change*. Put simply, the rate of environment change is the ratio of the speed of the agent's control loop to the rate of change of the environment. If the rate of world change is 1, then the environment will change no more than once for each time the agent can execute its control loop. If the rate of world change is 2, then the environment can change twice for each pass through the agent's control loop, and so on. The performance of an agent is measured by the ratio of number of intentions that the agent managed to achieve to the number of intentions that the agent had at any time. Thus if effectiveness is 1, then the agent achieved all its intentions. If effectiveness is 0, then the agent failed to achieve any of its intentions. The key results of Kinny and Georgeff were as follows.

- If the rate of world change is low (i.e. the environment does not change quickly), then bold agents do well compared with cautious ones. This is because cautious ones waste time reconsidering their commitments while bold agents are busy working towards – and achieving – their intentions.
- If the rate of world change is high (i.e. the environment changes frequently), then cautious agents tend to outperform bold agents. This is because they are able to recognize when intentions are doomed, and also to take advantage of serendipitous situations and new opportunities when they arise.

The bottom line is that different environment types require different intention reconsideration and commitment strategies. In static environments, agents that are strongly committed to their intentions will perform well. But in dynamic environments, the ability to react to changes by modifying intentions becomes more important, and weakly committed agents will tend to outperform bold agents.

4.4 HOMER: an Agent That Plans

An interesting experiment in the design of intelligent agents was conducted by Vere and Bickmore (1990). They argued that the enabling technologies for intelligent agents were sufficiently developed to be able to construct a prototype autonomous agent, with linguistic ability, planning and acting capabilities, and so on. They developed such an agent, and christened it HOMER. This agent is a simulated robot submarine, which exists in a two-dimensional 'Seaworld', about which it has only partial knowledge. HOMER takes instructions from a user in a limited subset of English with about an 800 word vocabulary; instructions can contain moderately sophisticated temporal references. HOMER can plan how to achieve its instructions (which typically relate to collecting and moving items around the

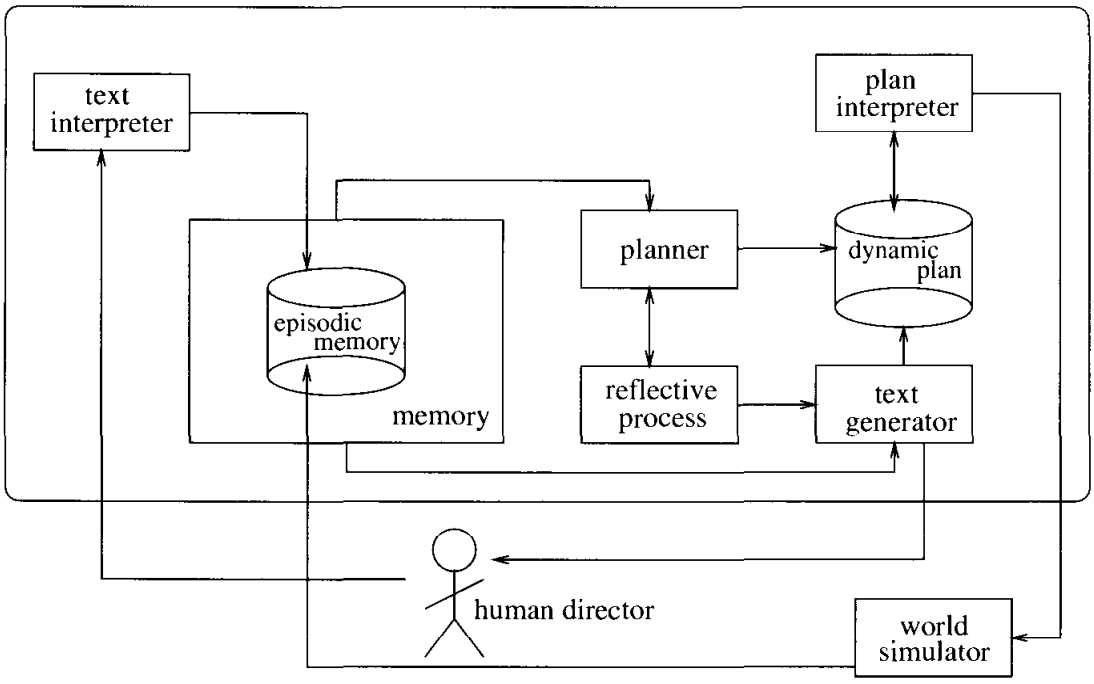

Figure 4.4 The HOMER architecture.

Seaworld), and can then execute its plans, modifying them as required during execution. The agent has a limited *episodic memory*, and using this, is able to answer questions about its past experiences. The architecture of HOMER is illustrated in Figure 4.4.

Here is an example of a dialogue with HOMER: the user begins by asking HOMER what he sees in front of him.

```
USER: What is in front of you?
HOMER: A log.
USER: Do you own the log.
HOMER: No, I don't.
```

The user then tells HOMER that he owns the log; this is new information, and so HOMER expresses surprise.

```
USER: The log belongs to you.
HOMER: Oh.
```

At this point, the user tells HOMER to move, so that he can no longer see the log, and moves the log while it is hidden from HOMER's sight. When he turns around, HOMER expects to see the log, and expresses surprise.

```
USER: Turn away from your log.
HOMER: OK, I am turning.
[LOG IS MOVED]
USER: Turn around.
HOMER: OK, I'm turning.
HOMER: I've lost the log!
```

In the following dialogue, the user gives HOMER a task to accomplish.

```
USER: Drop the package at the barge next Saturday at 9pm.
HOMER: OK.
```

HOMER then generates an appropriate plan for this task. When asked about his plans for Saturday, HOMER demonstrates that he has realized travelling to the barge will take some time.

```
USER: Are you going to the barge next Saturday?
HOMER: Yes.
USER: What time?
HOMER: 8.56pm.
```

4.5 The Procedural Reasoning System

The Procedural Reasoning System (PRS), originally developed at Stanford Research Institute by Michael Georgeff and Amy Lansky, was perhaps the first agent architecture to explicitly embody the belief–desire–intention paradigm, and proved to be the most durable agent architecture developed to date. It has been applied in several of the most significant multiagent applications so far built, including an air-traffic control system called OASIS that is currently undergoing field trials at Sydney airport, a simulation system for the Royal Australian Air Force called SWARMM, and a business process management system called SPOC (Single Point of Contact), that is currently being marketed by Agentis Solutions (Georgeff and Rao, 1996).

An illustration of the PRS architecture is given in Figure 4.5. The PRS is often referred to as a *belief–desire–intention* (BDI) architecture, because it contains explicitly represented data structures loosely corresponding to these mental states (Wooldridge, 2000b).

In the PRS, an agent does no planning from first principles. Instead, it is equipped with a library of pre-compiled plans. These plans are manually constructed, in advance, by the agent programmer. Plans in the PRS each have the following components:

- a *goal* – the postcondition of the plan;
- a *context* – the precondition of the plan; and
- a *body* – the 'recipe' part of the plan – the course of action to carry out.

The goal and context part of PRS plans are fairly conventional, but the body is slightly unusual. In the plans that we saw earlier in this chapter, the body of a plan was simply a sequence of actions. Executing the plan involves executing each action in turn. Such plans are possible in the PRS, but much richer kinds of plans are also possible. The first main difference is that as well has having individual

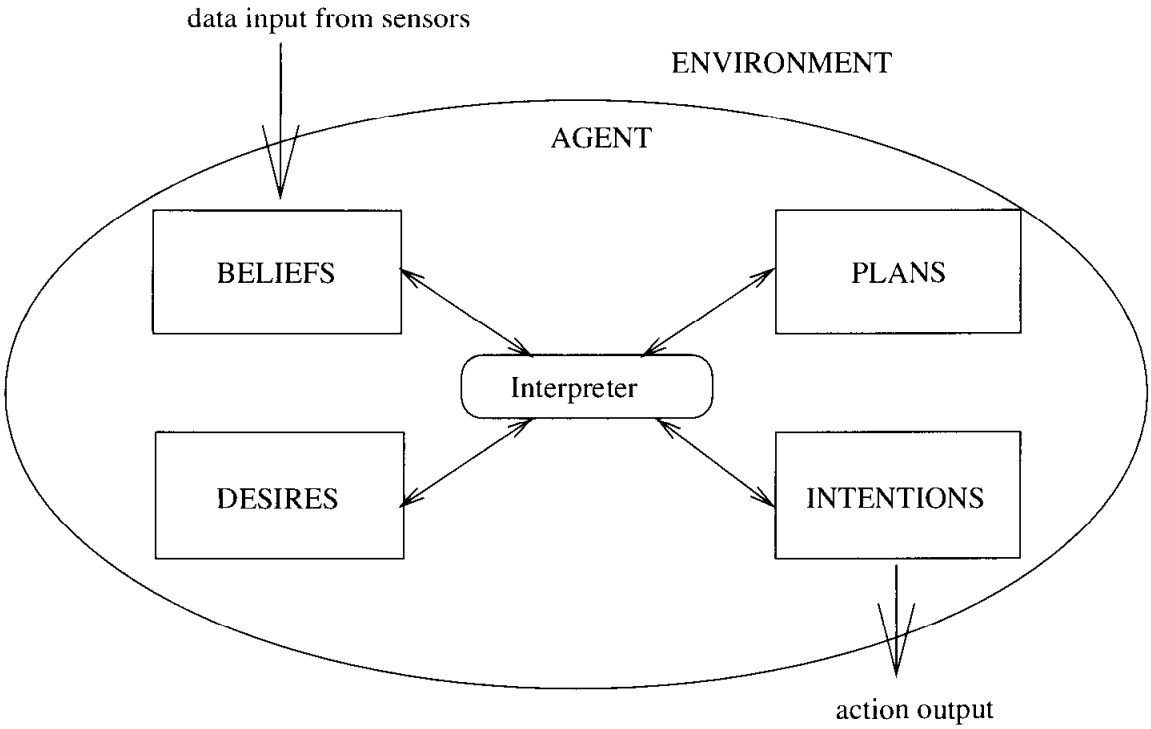

Figure 4.5 The Procedural Reasoning System (PRS).

primitive actions as the basic components of plans, it is possible to have *goals*. The idea is that when a plan includes a goal at a particular point, this means that this goal must then be achieved at this point before the remainder of the plan can be executed. It is also possible to have disjunctions of goals ('achieve φ or achieve ψ'), and loops ('keep achieving φ until ψ'), and so on.

At start-up time a PRS agent will have a collection of such plans, and some initial beliefs about the world. Beliefs in the PRS are represented as Prolog-like facts – essentially, as atoms of first-order logic, in exactly the same way that we saw in deductive agents in the preceding chapter. In addition, at start-up, the agent will typically have a top-level goal. This goal acts in a rather similar way to the 'main' method in Java or C.

When the agent starts up, the goal to be achieved is pushed onto a stack, called the *intention stack*. This stack contains all the goals that are pending achievement. The agent then searches through its plan library to see what plans have the goal on the top of the intention stack as their postcondition. Of these, only some will have their precondition satisfied, according to the agent's current beliefs. The set of plans that (i) achieve the goal, and (ii) have their precondition satisfied, become the possible *options* for the agent (cf. the $options$ function described earlier in this chapter).

The process of selecting between different possible plans is, of course, deliberation, a process that we have already discussed above. There are several ways of deliberating between competing options in PRS-like architectures. In the original PRS deliberation is achieved by the use of *meta-level plans*. These are literally

plans about plans. They are able to modify an agent's intention structures at run-time, in order to change the focus of the agent's practical reasoning. However, a simpler method is to use *utilities* for plans. These are numerical values; the agent simply chooses the plan that has the highest utility.

The chosen plan is then executed in its turn; this may involve pushing further goals onto the intention stack, which may then in turn involve finding more plans to achieve these goals, and so on. The process bottoms out with individual actions that may be directly computed (e.g. simple numerical calculations). If a particular plan to achieve a goal fails, then the agent is able to select another plan to achieve this goal from the set of all candidate plans.

To illustrate all this, Figure 4.6 shows a fragment of a Jam system (Huber, 1999). Jam is a second-generation descendant of the PRS, implemented in Java. The basic ideas are identical. The top level goal for this system, which is another Blocks World example, is to have achieved the goal `blocks_stacked`. The initial beliefs of the agent are spelled out in the FACTS section. Expressed in conventional logic notation, the first of these is $On(Block5, Block4)$, i.e. 'block 5 is on top of block 4'.

The system starts by pushing the goal `blocks_stacked` onto the intention stack. The agent must then find a candidate plan for this; there is just one plan that has this goal as a GOAL: the 'top level plan'. The context of this plan is empty, that is to say, true, and so this plan can be directly executed. Executing the body of the plan involves pushing the following goal onto the intention stack:

$$On(block3, table).$$

This is immediately achieved, as it is a FACT. The second sub-goal is then posted:

$$On(block2, block3).$$

To achieve this, the 'stack blocks that are clear' plan is used; the first sub-goals involve clearing both $block2$ and $block3$, which in turn will be done by two invocations of the 'clear a block' plan. When this is done, the `move` action is directly invoked to move $block2$ onto $block3$.

I leave the detailed behaviour as an exercise.

Notes and Further Reading

Some reflections on the origins of the BDI model, and on its relationship to other models of agency, may be found in Georgeff *et al.* (1999). Belief–desire–intention architectures originated in the work of the Rational Agency project at Stanford Research Institute in the mid-1980s. Key figures were Michael Bratman, Phil Cohen, Michael Georgeff, David Israel, Kurt Konolige, and Martha Pollack. The origins of the model lie in the theory of human practical reasoning developed by the philosopher Michael Bratman (Bratman, 1987), which focuses particularly on

```
GOALS:
  ACHIEVE blocks_stacked;
FACTS:
  FACT ON "Block5" "Block4";      FACT ON "Block4" "Block3";
  FACT ON "Block1" "Block2";      FACT ON "Block2" "Table";
  FACT ON "Block3" "Table";       FACT CLEAR "Block1";
  FACT CLEAR "Block5";            FACT CLEAR "Table";
Plan: {
  NAME: "Top-level plan"
  GOAL: ACHIEVE blocks_stacked;
  CONTEXT:
  BODY:     ACHIEVE ON "Block3" "Table";
            ACHIEVE ON "Block2" "Block3";
            ACHIEVE ON "Block1" "Block2";
}
Plan: {
  NAME: "Stack blocks that are already clear"
  GOAL: ACHIEVE ON $OBJ1 $OBJ2;
  CONTEXT:
  BODY:     ACHIEVE CLEAR $OBJ1;
            ACHIEVE CLEAR $OBJ2;
            PERFORM move $OBJ1 $OBJ2;
  UTILITY: 10;
  FAILURE: EXECUTE print "\n\nStack blocks failed!\n\n";
}
Plan: {
  NAME: "Clear a block"
  GOAL: ACHIEVE CLEAR $OBJ;
  CONTEXT: FACT ON $OBJ2 $OBJ;
  BODY:     ACHIEVE ON $OBJ2 "Table";
  EFFECTS: RETRACT ON $OBJ2 $OBJ;
  FAILURE: EXECUTE print "\n\nClearing block failed!\n\n";
}
```

Figure 4.6 The Blocks World in Jam.

the role of intentions in practical reasoning. The conceptual framework of the BDI model is described in Bratman *et al.* (1988), which also describes a specific BDI agent architecture called IRMA.

The best-known implementation of the BDI model is the PRS system, developed by Georgeff and colleagues (Georgeff and Lansky, 1987; Georgeff and Ingrand, 1989). The PRS has been re-implemented several times since the mid-1980s, for example in the Australian AI Institute's DMARS system (d'Inverno *et al.*, 1997), the University of Michigan's C++ implementation UM-PRS, and a Java version called

Jam! (Huber, 1999). Jack is a commercially available programming language, which extends the Java language with a number of BDI features (Busetta *et al.*, 2000).

The description of the BDI model given here draws upon Bratman *et al.* (1988) and Rao and Georgeff (1992), but is not strictly faithful to either. The most obvious difference is that I do not incorporate the notion of the 'filter override' mechanism described in Bratman *et al.* (1988), and I also assume that plans are linear sequences of actions (which is a fairly 'traditional' view of plans), rather than the hierarchically structured collections of goals used by PRS.

Plans are central to the BDI model of agency. An excellent discussion on the BDI model, focusing in particular on the role of plans in practical reasoning, is Martha Pollack's 1991 *Computers and Thought* award lecture, presented at the IJCAI-91 conference in Sydney, Australia, and published as 'The Uses of Plans' (Pollack, 1992). Another article, which focuses on the distinction between 'plans as recipes' and 'plans as mental states' is Pollack (1990). It is worth emphasizing that the BDI model is only one solution to the problem of building autonomous rational agents. Many other software architectures for agent systems have been described in the literature (Wooldridge and Jennings, 1995; Brooks, 1999). Other practical reasoning-style architectures include Fischer *et al.* (1996), Jung (1999), Móra *et al.* (1999) and Busetta *et al.* (2000).

The BDI model is also interesting because a great deal of effort has been devoted to formalizing it. In particular, Anand Rao and Michael Georgeff have developed a range of BDI logics, which they use to axiomatize properties of BDI-based practical reasoning agents (Rao and Georgeff, 1991a; Rao *et al.*, 1992; Rao and Georgeff, 1991b; Rao and Georgeff, 1992; Rao and Georgeff, 1993; Rao, 1996b). These models have been extended by others to deal with, for example, communication between agents (Haddadi, 1996).

Class reading: Bratman *et al.* (1988). This is an interesting, insightful article, with not too much technical content. It introduces the IRMA architecture for practical reasoning agents, which has been very influential in the design of subsequent systems.

Exercises

(1) **[Level 1.]**

Imagine a mobile robot, capable of moving around an office environment. Ultimately, this robot must be controlled by very low-level instructions along the lines of 'motor on', and so on. How easy would it be to develop STRIPS operators to represent these properties? Try it.

(2) **[Level 2.]**

Recall the vacuum-world example discussed in the preceding chapter. Formulate the operations available to the agent using the STRIPS notation.

(3) **[Level 2.]**

Consider an agent that must move from one location to another, collecting items from one site and moving them. The agent is able to move by taxi, bus, bicycle, or car.

Formalize the operations available to the agent (move by taxi, move by car, etc.) using the STRIPS notation. (Hint: preconditions might be having money or energy.)

(4) **[Level 3.]**

Read Kinny and Georgeff (1991), and implement these experiments in the programming language of your choice. (This is not as difficult as its sounds: it should be possible in a couple of days at most.) Now carry out the experiments described in Kinny and Georgeff (1991) and see if you get the same results.

(5) **[Level 3.]**

Building on the previous question, investigate the following.

The effect that reducing perceptual capabilities on agent performance. The idea here is to reduce the amount of environment that the agent can see, until it can finally see only the grid square on which it is located. Can 'free' planning compensate for the inability to see very far?

The effect of non-deterministic actions. If actions are allowed to become non-deterministic (so that in attempting to move from one grid square to another, there is a certain probability that the agent will in fact move to an entirely different grid square), what effect does this have on the effectiveness of an agent?