BDI Architecture

The *belief-desire-intention* (BDl) architectures have their roots in the philosophical tradition of understanding *practical reasoning* – the process of deciding, moment by moment, which action to perform in the furtherance of our goals.

Practical reasoning involves two important processes: deciding *what* goals we want to achieve, and *how* we are going to achieve these goals. The former process is known as *deliberation,* the latter as *means-ends* reasoning.

The process of practical reasoning in a *BDI* agent is summarized in figure 1. As this figure illustrates, there are seven main components to a *BDI* agent:

- a set of current *beliefs,* representing information the agent has about its current environment;
- a *belief revision function,* (*brf*), which takes a perceptual input and the agent's current beliefs, and on the basis of these, determines a new set of beliefs;
- an *option generation function,* (*options*), which determines the options available to the agent (its desires), on the basis of its current beliefs about its environment and its current *intentions;*
- a set of *current options,* representing possible courses of actions available to the agent;
- a *filter* function (*filter*), which represents the agent's *deliberation* process, and which determines the agent's intentions on the basis of its current beliefs, desires, and intentions;
- a set of current *intentions,* representing the agent's current focus – those states of affairs that it has committed to trying to bring about;
- an *action selection function* (*execute*), which determines an action to perform on the basis of current intentions.

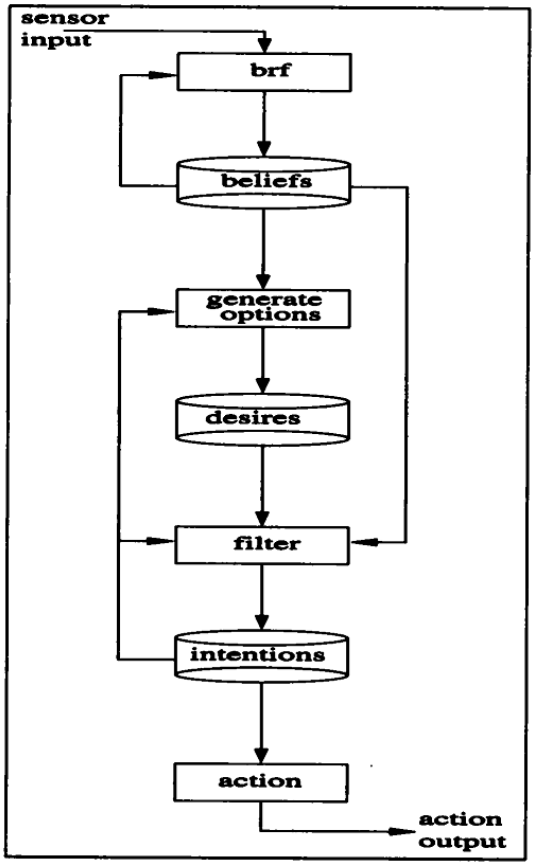

Figure 1. *Schematic diagram of a generic belief-desire-intention architecture*

It is straightforward to formally define these components. First, let *Bel* be the set of all possible beliefs, *Des* be the set of all possible desires, and *Int* be the set of all possible intentions. For the purposes of this chapter, the content of these sets is not important. (Often, beliefs, desires, and intentions are represented as logical formulae, perhaps of first-order logic.) Whatever the content of these sets, it is worth noting that they should have some notion of *consistency* defined upon them, so that one can answer the question of, for example, whether having an intention to achieve x is consistent with the belief that y. Representing beliefs, desires, and intentions as logical formulae permits us to cast such questions as questions of determining whether logical formulae are consistent – a well-known and well-understood problem. The state of a BDI agent at any given moment is,

unsurprisingly, a triple (B, *D, I*), where $B \subseteq Bel$, $D \subseteq Des$, and $I \subseteq Int$. An agent's belief revision function is a mapping:

$$brf : \wp(Bel) \times P \rightarrow \wp(Bel),$$

which on the basis of the current percept and current beliefs determines a new set of beliefs. Belief revision is out of the scope of this chapter and so we shall say no more about it here.

The option generation function, *options*, maps a set of beliefs and a set of intentions to a set of desires:

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

This function plays several roles. First, it must be responsible for the agent's means-ends reasoning – the process of deciding how to achieve intentions. Thus, once an agent has formed an intention to *x*, it must subsequently consider options to *achieve x*. These options will be more concrete – less abstract – than *x*. As some of these options then become intentions themselves, they will also feedback into option generation, resulting in yet more concrete options being generated. We can thus think of a BDI agent's option generation process as one of recursively elaborating a hierarchical plan structure, considering and committing to progressively more specific intentions, until finally it reaches the intentions that correspond to immediately executable actions.

While the main purpose of the *options* function is thus means-ends reasoning, it must in addition satisfy several other constraints. First, it must be *consistent:* any options generated must be consistent with both the agent's current beliefs and current intentions. Secondly, it must be *opportunistic*, in that it should recognize when environmental circumstances change advantageously, to offer the agent new ways of achieving intentions, or the possibility of achieving intentions that were otherwise unachievable.

A BDI agent's deliberation process (deciding *what* to do) is represented in the *filter* function,

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

which updates the agent's intentions on the basis of its previously-held intentions and current beliefs and desires. This function must fulfil two roles. First, it must *drop* any intentions that are no longer achievable, or for which the expected cost of achieving them exceeds the expected gain associated with successfully achieving them. Second, it should retain intentions that are not achieved, and that are still expected to have a positive overall benefit. Finally, it should *adopt* new intentions, either to achieve existing intentions, or to exploit new opportunities.

Notice that we do not expect this function to introduce intentions from nowhere. Thus *filter* should satisfy the following constraint:

$$\forall B \in \wp(Bel),\ \forall D \in \wp(Des),\ \forall I \in \wp(Int),\ filter(B, D, I) \subseteq I \cup D.$$

In other words, current intentions are either previously held intentions or newly adopted options.

The *execute* function is assumed to simply return any executable intentions – one that corresponds to a directly executable action:

$$execute : \wp(Int) \rightarrow A.$$

The agent decision function, *action* of a BDI agent is then a function:

$$action : P \rightarrow A,$$

and is defined by the following pseudo-code:

```
1.  function action(p : P) : A
2.  begin
3.      B := brf(B, p)
4.      D := options(D, I)
5.      I := filter(B, D, I)
6.      return execute(I)
7.  end function action
```

Note that representing an agent's intentions as a *set* (i.e., as an unstructured collection) is generally too simplistic in practice. A simple

alternative is to associate a *priority* with each intention, indicating its relative importance. Another natural idea is to represent intentions as a *stack.* An intention is pushed on to the stack when it is adopted, and popped when it is either achieved or else not achievable. More abstract intentions will tend to be at the bottom of the stack, with more concrete intentions towards the top.

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*(...) function which requires an agent to engage in *plan generation* – constructing a plan from. In many implemented practical reasoning agents, the *plan*(...) function is implemented by giving the agent a *plan library*. 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.

We can now discuss the overall control structure of a practical reasoning agent. Figure 2 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:

- 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.

```
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 2. *A practical reasoning agent*

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).

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?

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:

- *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 2 is as follows. An agent will maintain a commitment to an intention until:

- it believes the intention has succeeded;
- it believes the intention is impossible; or
- there is nothing left to execute in the plan.

This is single-minded commitment. T 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 slop to *reconsider* its intentions? One possibility is to reconsider intentions at every opportunity – in particular, alter 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 2 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.

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.

To summarize, BDI architectures are practical reasoning architectures, in which the process of deciding what to do resembles the kind of practical reasoning that we appear to use in our everyday lives. The basic components of a BDI architecture are data structures representing the beliefs, desires, and intentions of the agent, and functions that represent its deliberation (deciding what intentions to have – i.e., deciding what to do) and means-ends reasoning (deciding how to do it). Intentions play a central role in the BDI model: they provide stability for decision making, and act to focus the agent's practical reasoning. A major issue in BDI architectures is the problem of striking a balance between being committed to and

overcommitted to one's intentions: the deliberation process must be finely tuned to its environment, ensuring that in more dynamic, highly unpredictable domains, it reconsiders its intentions relatively frequently – in more static environments, less frequent reconsideration is necessary.

The BDI model is attractive for several reasons. First, it is intuitive – we all recognize the processes of deciding what to do and then how to do it, and we all have an informal understanding of the notions of belief, desire, and intention. Second, it gives us a clear functional decomposition, which indicates what sorts of subsystems might be required to build an agent. But the main difficulty, as ever, is knowing how to efficiently implement these functions.

References

M. Wooldridge, *Intelligent Agents*, in G. Weiss (ed.), Multiagent systems, The MIT Press, 1999.

M. Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons, 2002.

A. S. Rao, M. P. Georgeff, *Modeling rational agents within a BDI-architecture*, in R. Fikes, E. Sandewall (eds.), Proceedings of Knowledge Representation and Reasoning (KR&R-91), pp. 473-484, Morgan Kaufmann Publishers, San Mateo, CA, USA, 1991.

M. P. Georgeff, A. L. Lansky, *Reactive reasoning and planning*, Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI -87), pp. 677 -682, Seattle, WA, USA, 1987.

D. Kinny, M. Georgeff, *Commitment and effectiveness of situated agents*, Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI -91), pp. 82-88, Sydney, Australia, 1991.