Reasoning about BDI Agents from a Programming Languages Perspective

Wayne Wobcke
School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
wobcke@cse.unsw.edu.au

Abstract

In this paper, we summarize an approach to reasoning about a class of BDI agent architectures based on PRS. The theory is formalized using a logic, Agent Dynamic Logic (ADL), that combines elements from Computation Tree Logic, Propositional Dynamic Logic and Rao and Georgeff's BDI Logic. The motivation of this work is to develop a logical framework that is at once rigorous in providing formal notions of belief, desire and intention, yet which is also computationally grounded in the operational behaviour of this architecture, so as to enable formal reasoning about the behaviour of agents in this class. We illustrate the model theory with a simple "waypoint following" agent.

Methodology

Following a symposium on intentions in communication almost exactly twenty years ago, papers that were eventually published as Bratman (1990) and Cohen and Levesque (1990b) established a research direction in the interaction between philosophical and formal logic theories of intention and action, explicitly recognized in Allen's commentary (Allen, 1990). Perhaps the main issue can be concisely stated as to provide a general formal modelling of intention and action (and their relationship) for rational agents that is consistent with philosophical theories of intention, action and rationality. Such a theory would enable reasoning about rational agents using a logical approach, possibly even to prove the rationality of complex "BDI agents", those based on notions of belief, desire and intention.

The problem of developing a general logical theory of intention and rationality for BDI agents, however, remains open. We believe that part of the difficulty in developing such a general theory is that formal modellings inevitably build in some architectural assumptions about the agents they model, so inevitably a formal theory applies only to certain classes of BDI agents. Moreover, when it comes to reasoning about agents in that class, what is required is a systematic mapping from the computational states of the agent to formal BDI models. This requirement is what Wooldridge (2000) has called *computational grounding*. If a formal theory is not computationally grounded, properties of an agent's mental states that are shown to hold in virtue of some formal modelling do not necessarily apply to the agent, making the "cognitive" analysis of the agent irrelevant for practical considerations.

Our approach is therefore to start by developing a more specific logical account of intention and action for the PRS-style agent architecture, Georgeff and Lansky (1987), adopting a point of view similar to that used in reasoning about computer programming languages. The PRS-type architecture operates with some notions of belief, desire and intention, and is accordingly described as a "BDI architecture". However using logical versions of idealized concepts of belief, desire and intention to model this architecture is inadequate for reasoning about agent programs, because these idealized concepts do not match their specific meanings as used in the architecture. Rather, the "PRS-like" agent uses a plan library to store pre-defined plans for achieving goals, explicit beliefs to represent information about the environment that guides the selection of actions both in response to events and to refine plans, and various strategies for action execution to realize commitments to the chosen plans.

Although our work focuses on the properties of a particular BDI architecture, our aim is to develop the logic of intention and action in a general way so as not to be restricted to agents of this class. In particular, our clause for the semantics of intention is based on Bratman's requirement of strong consistency between intentions and beliefs, and is not specific to any architecture. However, our approach to modelling the agent's action and reasoning incorporates two assumptions particular to PRS: (i) that the agent attempts to execute only one primitive action at a time, and (ii) that the agent selects a plan to fulfil an achievement goal only at the time of executing that plan (at the latest possible moment). These assumptions reflect more the simplicity of PRS and/or the characteristics of the environments in which PRS agents operate, rather than general principles of rationality.

The organization of this paper is as follows. We begin with a brief summary of the PRS-like agent architecture, then describe some of the philosophical concerns that have been addressed in our approach, present our logic, Agent Dynamic Logic (ADL), for reasoning about intention and action, and finally, conclude with an illustration of the computational grounding of PRS-like agents based on a simple "waypoint following" agent.

PRS-Like Agent Architectures

The class of agent architectures studied in this paper are the *PRS-like* architectures, which is supposed to cover PRS (Procedural Reasoning System), Georgeff and Lansky (1987), and variants such as UM-PRS, C-PRS, AgentSpeak(L), dMARS, JAM, JACK Intelligent Agents^{TM} and SPARK. The agent's computation cycle can be conveniently described with reference to the simplified interpreter shown in Figure 1, adapted from Rao and Georgeff (1992). In this abstract interpreter, the system state consists of a set of beliefs B and intentions I. Each element of I is a partially executed hierarchical plan.

```
Abstract BDI Interpreter:
    initialize-state(B, I);
    do
        get-external-event(e);
        new-options := trigger-plans(e, B);
        selected-option := select-option(new-options);
        update-intentions(selected-option, I);
        selected-intention := select-intention(I);
        execute(selected-intention);
        update-intention(selected-intention, I);
        get-observation(o);
        update-beliefs(o, B);
        drop-successful-plans(B, I);
        drop-impossible-plans(B, I)
    until quit
```

Figure 1: Abstract BDI Interpreter

The *configurations* of the agent are pairs $\langle \mathsf{B}, \mathsf{I} \rangle$ where B is the set of beliefs and I is the set of intentions (partially executed hierarchical plans). We assume a finite propositional language $\mathcal{L}$ for representing the beliefs of the agent, and the agent's belief set B at any time is assumed to be a consistent set of literals of $\mathcal{L}$. The conditions of each plan in the plan library are also formulae of $\mathcal{L}$, and the body of each plan is a program. The language of programs consists of a set of atomic programs, including special actions *achieve* γ (where γ is a formula of $\mathcal{L}$) and an "empty" program Λ, and conditional and iterative statements **if** α **then** π **else** ψ and **while** α **do** π (where α is a formula of $\mathcal{L}$ and π and ψ are programs). Note that the tests in these statements are tests on the agent's beliefs, not on the state of the environment.

The selection mechanisms are constrained as follows:

- Given a set of options triggered by an event (whose precondition and context are believed), the function select-option returns a randomly selected element of maximal priority within that set;
- Given a set of intentions, the function select-intention returns a randomly selected element of maximal priority in the subset of this set of plans which are applicable (whose preconditions are believed) in the current state; moreover, if a plan is chosen to break a tie, as long as its execution is believed successful, it continues to be chosen on future cycles until it terminates.

The functions for belief and intention update are as follows. For simplicity, it is assumed that the observations on each cycle correspond to a consistent set of literals O. Then the belief revision function can be defined as the function that removes the complement $\bar{l}$ of each literal l in O from the belief set B (if it is contained in B) and then adds each (now consistent) literal l of O to B. The only subtlety with intention update is in defining which plans may be dropped as achieved or infeasible. We take it that on failure, only whole plans can be dropped, and on success, only whole plans or the initial actions of any plan can be dropped (if an action of the form *achieve* γ is dropped, the achieve action and all its subplans are removed from the intention structure). Dropping a subplan leaves the achieve subgoal that resulted in the plan's selection as the next step to be executed in that hierarchical plan.

Philosophical Issues

Any formalism for reasoning about BDI agents makes concrete some implicit theory of intention, and since one major aim of formalization is to give a semantic definition of intention in terms of certain model theoretic constructs, any formalism implicitly includes a partial theory of intention. There are a number of philosophical issues, listed below, dealing with notions of agency, intentions vs. intentional actions, actions vs. events, ability and control, that must be handled correctly in any formalism and computational approach applying to BDI agents. These issues do not always form part of the formalism; sometimes they relate to assumptions about how the formalism is meant to be applied in reasoning about agents.

The main definitions in any formalism relate intention to other concepts, such as goals and events, in the case of Cohen and Levesque (1990a), or actions, in the case of Rao and Georgeff (1991) and Singh (1998). In our approach, a version of Bratman's requirement for strong consistency on beliefs and intentions is the basis of the modelling of intention. This requirement is never defined precisely by Bratman; the closest statement approaching a definition is that an agent's intentions are *strongly consistent relative to its beliefs* when it is 'possible for [its] plans taken together to be successfully executed in a world in which [its] beliefs are true', Bratman (1990, p. 19), though Bratman realizes that this is in need of further elaboration, Bratman (1987, p. 179, note 3). It is indeed a strong requirement, since the quantification in 'a world in which its beliefs are true' seems to refer to *all* worlds in which the agent's beliefs are true, not just some such worlds or the actual world. In our approach, rather than taking all worlds in which the agent's beliefs are true (which is inappropriate for simple PRS-type agents with a limited belief language), we define intention with respect to a set of all worlds the agent could inhabit (at the time of assigning the intention), which, moreover, are determined at any time from the prior execution of the agent's program (however, typically starting from a situation satisfying the agent's beliefs). Branching in such structures is derived from the nondeterministic execution of action attempts, so some branches correspond to successful executions while others correspond to failed attempts, and the intentions of the agent are directed towards the successful performance of those actions the agent chooses from its plans.

Our formal approach to modelling belief and intention for PRS-like agents aims to address the following aspects of belief, intention and action:

- *Present vs. future-directed intentions:* Intentions are future-directed and directed towards actions, not propositions (so Bratman's idea of intentions "controlling conduct" is captured, though the actions attempted by the agent are assumed to be under its control);
- *Success and failure:* Intentions are directed towards actions the agent will eventually perform successfully in every world considered possible (so a rationality assumption of belief-intention consistency is built in, though not a requirement for successful fulfilment of the intention since there can be unintended failed action attempts);
- *Action attempts vs. events:* By treating attempts as a type of action, the model theory captures a distinction between what the agent actually does and what the agent tries to do (intentions are directed towards successful execution of the agent's attempts, whereas Rao and Georgeff (1991) have intentions directed towards "events" that may or may not succeed);
- *Two types of success:* The use of attempts and the semantics for *achieve* γ actions enable the theory to capture the distinction between an agent successfully executing the steps in a plan and successfully achieving the postcondition and goal (intended effect) of the plan;
- *Ignorance vs. nondeterminism:* Ignorance is modelled by a belief-alternative relation between situations in worlds, nondeterminism through branching within worlds (this differs from Rao and Georgeff's motivation based on decision theory);
- *Beliefs in the action semantics:* To correctly handle actions that test the agent's beliefs and the fact that agents "track" conditions about the environment brought about by their actions, belief relations are included in the semantics of action, in that an action is modelled as a relation on environment and associated belief states;
- *Two types of "possible worlds":* The modelling distinguishes worlds the agent could inhabit (alternative actual worlds) from the beliefs (epistemic possibilities) of the agent in those worlds (intention relates to the former, belief to the latter);
- *Side-effect or "package deal" problems:* The account handles side-effect problems in an intuitive way, in that an agent can intend to perform an action but not its generated consequences, since the plans for the consequences can differ from the plan the agent has adopted, and in addition since an agent "tracks" through observation only the intended consequences of its actions.

Agent Dynamic Logic

In this section, we summarize our formal framework for modelling BDI agents, which is based on a logic called Agent Dynamic Logic (ADL), Wobcke (2002), that combines aspects of Computation Tree Logic (CTL), Emerson and Clarke (1982), Propositional Dynamic Logic (PDL), Pratt (1976), and the BDI Logic of Rao and Georgeff (1991), called here BDI-CTL.

The language of ADL (Agent Dynamic Logic) is based on both BDI-CTL, which extends CTL with modal operators for modelling beliefs, desires (goals) and intentions, and PDL, which includes modal operators corresponding to program terms. Our definitions of BDI-CTL are modifications of Rao and Georgeff's in that, though there are three modal operators, B (belief), G (goal) and I (intention) in the language, the operators G and I are defined in terms of other primitives. We assume there is a base propositional language $\mathcal{L}$ for expressing time-independent properties of states. The language of ADL includes the formulae of BDI-CTL (which includes the CTL state formulae) plus formulae built using the PDL constructs $*$ (iteration), ; (sequencing) and $\cup$ (alternation), and, corresponding to any formula α of the base language $\mathcal{L}$, a test statement $\alpha?$. Note, however, that the test statement covers tests on the beliefs of the agent in addition to tests on the environment. The language of programs also includes special atomic actions *achieve* γ, *attempt* π and *observe* α (where γ and α are formulae of the base language $\mathcal{L}$ with α a conjunction of literals, and π is a program).

Rao and Georgeff (1991) make extensive use of the notion of a "world", a forward-branching discrete-time structure.

Definition 1 *Let $\mathcal{S}$ be a nonempty set of states. A* world w *over a time tree $\langle \mathcal{T}, \prec \rangle$ based on $\mathcal{S}$ is a function on $\mathcal{T}$ giving a state $w_t \in \mathcal{S}$ for each time point $t \in \mathcal{T}$ (in the context of a particular world w, w_t is called a* situation*).*

Definition 2 *A* BDI interpretation $\langle \mathcal{T}, \prec, \mathcal{S}, \mathcal{W}, \mathcal{A}, \mathcal{B}, \mathcal{I} \rangle$ *is a tuple where $\langle \mathcal{T}, \prec \rangle$ is a time tree, $\mathcal{S}$ is a nonempty set of states, $\mathcal{W}$ is a nonempty set of worlds based on $\mathcal{S}$, each over a subtree of $\langle \mathcal{T}, \prec \rangle$, $\mathcal{A}$ and $\mathcal{B}$ are subsets of $\mathcal{W} \times \mathcal{T} \times \mathcal{W} \times \mathcal{T}$ defined only for tuples (w, t, w', t') for which t (t') is a time point in w (w') and w_t and $w'_{t'}$ share a common history, and $\mathcal{I}$ is a function $\mathcal{W} \times \mathcal{T} \rightarrow \mathcal{W}$ mapping each time point t in a world w to a subworld of w containing t.*

Definition 3 *A* subworld *of a world w over a time tree $\langle \mathcal{T}, \prec \rangle$ based on a set of states $\mathcal{S}$ is the world w restricted to a subtree of $\langle \mathcal{T}, \prec \rangle$ whose root is the root of w.*

As mentioned above, the agent needs to keep track of two senses of epistemic alternative: alternative actualities and alternative epistemic states with respect to one actual world. The relation $\mathcal{A}$ is for alternative actual worlds, while the relation $\mathcal{B}$ captures the ignorance of an agent with respect to one actual world. The relation $\mathcal{A}$ is used as part of the definition of intentions; no matter how the world is actually, the agent will eventually perform the intended action on all possible futures which it regards as successfully realizing its intentions. Both relations $\mathcal{A}$ and $\mathcal{B}$ on situations are assumed to be serial, transitive and Euclidean, and in addition $\mathcal{A}$ is assumed to be reflexive.

The major definitions in ADL relate to the modal operators for belief, desire (goal) and intention.

Definition 4 *Let $\langle \mathcal{T}, \prec, \mathcal{S}, \mathcal{W}, \mathcal{A}, \mathcal{B}, \mathcal{I} \rangle$ be a BDI interpretation. Then a world $w \in \mathcal{W}$ satisfies a* BDI-CTL *formula at a time point t in w as follows.*
$w \models_t \mathsf{B}\alpha$ *if* $w' \models_{t'} \alpha$ *whenever* $\mathcal{B}(w, t, w', t')$
$w \models_t \mathsf{I}\pi$ *if* $\mathcal{I}(w', t') \models_{t'} \forall\Diamond do(\pi)$ *whenever* $\mathcal{A}(w, t, w', t')$
$w \models_t \mathsf{G}\gamma$ *if* $w \models_t \mathsf{I}(achieve\ \gamma)$

A main feature of our approach is the semantic definition of intention, which explicitly relates intentions to actions eventually performed successfully (as defined by the function $\mathcal{I}$) in all alternative worlds the agent could actually inhabit (as determined by the relation $\mathcal{A}$). This captures the intuition that an agent intends to execute its actions successfully in the future in all of a range of possible worlds, though successful performance of the action is not guaranteed. The definition therefore accommodates the asymmetry thesis of Bratman (1987). The definition for goals, defining goals in terms of the postconditions of plans, is specific to PRS-type agents, which have no separate representation of goals.

We can now present the dynamic part of ADL. Analogous to PDL, the language of ADL includes modal operators $[\pi]$ and $\langle\pi\rangle$ corresponding to each program π, and the semantics of action is based on agent computation trees, extending the approach of Harel (1979). For PRS-type agents, the semantics of action can be defined without reference to intention and intention update, so the relations $\mathcal{A}$ and $\mathcal{I}$ play no role and are omitted. A situation in such an agent computation tree thus corresponds to a state of the environment together with the agent's set of beliefs.

The semantics of ADL is based on *PRS interpretations*. Each program is modelled as a set of agent computation trees that arises by varying the initial situation. Each agent computation tree b modelling a program π is a set of worlds, one distinguished world b^* modelling the execution of π in the environment and other worlds for representing the epistemic alternatives of each situation in b^*: transitions in such worlds correspond to "actions" of the form *observe* α where α is a conjunction of literals which implies, for each literal in the postcondition of π, either that literal or its complement. The world b^* may have *non-final* situations, situations in the agent computation tree where execution can continue. Each primitive action π is modelled as a set of agent computation trees of depth 1; the agent computation trees for complex programs are derived from these.

Definition 5 *A* PRS interpretation *is a pair $\langle\mathcal{S},\mathcal{R}\rangle$, where $\mathcal{S}$ is a set of states and $\mathcal{R}$ is a set of agent computation trees $\mathcal{R}_\pi$ based on $\mathcal{S}$, one such set for each program π, for which for each primitive program π, the set of transitions for $\mathcal{R}_\pi$ is a subset of those for $\mathcal{R}_{attempt\ \pi}$.*

The program construction operators, sequencing, alternation and iteration, are modelled as operations on agent computation trees. The operation for sequencing is a kind of "concatenation" of worlds $\oplus$, analogous to concatenation of computation sequences. For alternation, we utilize an operation $\uplus$ that merges two worlds if they have equivalent initial situations (two situations are *equivalent* if they represent the same state of the environment and beliefs of the agent). Iteration is modelled as a transitive closure operation over $\oplus$.

Definition 6 *The sets of agent computation trees $\mathcal{R}_{achieve\ \gamma}$ for the actions achieve γ are defined as follows.*

$$\mathcal{R}_{achieve\ \gamma}(\sigma) = \biguplus\{\mathcal{R}_\pi|_\gamma : \pi \in L_\gamma(\sigma)\}$$

Here $\mathcal{R}_{achieve\ \gamma}(\sigma)$ is the agent computation tree with root σ, $w|_\gamma$ is defined as the maximal subworld of w all of whose non-final situations satisfy γ, and $L_\gamma(\sigma)$ is the set of maximal priority plans from the plan library L amongst those applicable at σ and whose postcondition implies γ.

Finally, the satisfaction conditions for $do(\pi)$ and $[\pi]\alpha$ are defined as follows. Note that the definition for $do(\pi)$ is future-directed, in that $do(\pi)$ is a CTL state formula capturing a range of possible outcomes when executed in a situation. Thus $do(\pi)$ captures the choice of actions available to the agent at a point in time. The formula $[\pi]\alpha$ is quite different, stating that whenever π actually occurs in a situation, α holds in the resulting situation.

Definition 7 *Let $\langle\mathcal{S},\mathcal{R}\rangle$ be a PRS interpretation and let $\langle\mathcal{T},\prec,\mathcal{S},\mathcal{W},\mathcal{A},\mathcal{B},\mathcal{I}\rangle$ be a BDI interpretation. For a world $w\in\mathcal{W}$ over the time tree $\langle\mathcal{T},\prec\rangle$ containing a time point t, let w^t be the subworld of w over $\langle T_t,\prec_t\rangle$, the subtree of $\langle T,\prec\rangle$ generated from t. Then w satisfies $do(\pi)$ and $[\pi]\alpha$ at a time point t in w as follows.*

$w \models_t do(\pi)$ *if some prefix of w^t is isomorphic to a subworld of b^*, where b is the agent computation tree in $\mathcal{R}_\pi$ whose root is equivalent to w_t*

$w \models_t [\pi]\alpha$ *if some prefix of a subworld of w^t is isomorphic to a subworld of b^*, where b is the agent computation tree in $\mathcal{R}_\pi$ whose root is equivalent to w_t, and $w \models_u \alpha$ for every situation w_u in w corresponding to a non-final situation in b^**

Computational Grounding

In this section, we illustrate the modelling of intention and action using agent computation trees through an example "waypoint following" agent. As in standard approaches to both temporal logic and model checking, instead of working directly with agent computation trees, we use agent computation graphs, which can be "unwound" to form trees. These graphs are similar to the reachability graphs used in the SPIN model checker, Holzmann (1997), which represent all situations that can be reached from a given initial situation.

Each situation in an agent computation graph is associated with both a state of the environment s (including possibly an event e that occurs in the state) and a configuration of the agent, which consists of two sets B and I, where B is the set of beliefs and I the set of intentions of the agent in that situation. Situations representing epistemic alternatives of the agent are, in addition, associated with a state of the environment s' consistent with the agent's beliefs. Situations are only identified if all these values are identical. Note that an attempt is considered "failed" if its postcondition does not hold in, and importantly, if some (unexpected) event occurs in, the resulting situation.

Let us first present at the waypoint agent example before returning to more general discussion. The waypoint agent has a simple task: it must visit four waypoints, numbered 1–4, in order, whilst not running out of fuel. The agent is capable of carrying out the actions $visit_i$ and $refuel$; the refuelling plans are used in response to a $warn$ event in order to direct the agent to a fuel depot where a $refuel$ action is attempted. There is fuel at locations 1 and 3 though this knowledge is not explicit in the agent's beliefs. Rather, the agent is constructed to have four refuelling plans (two for

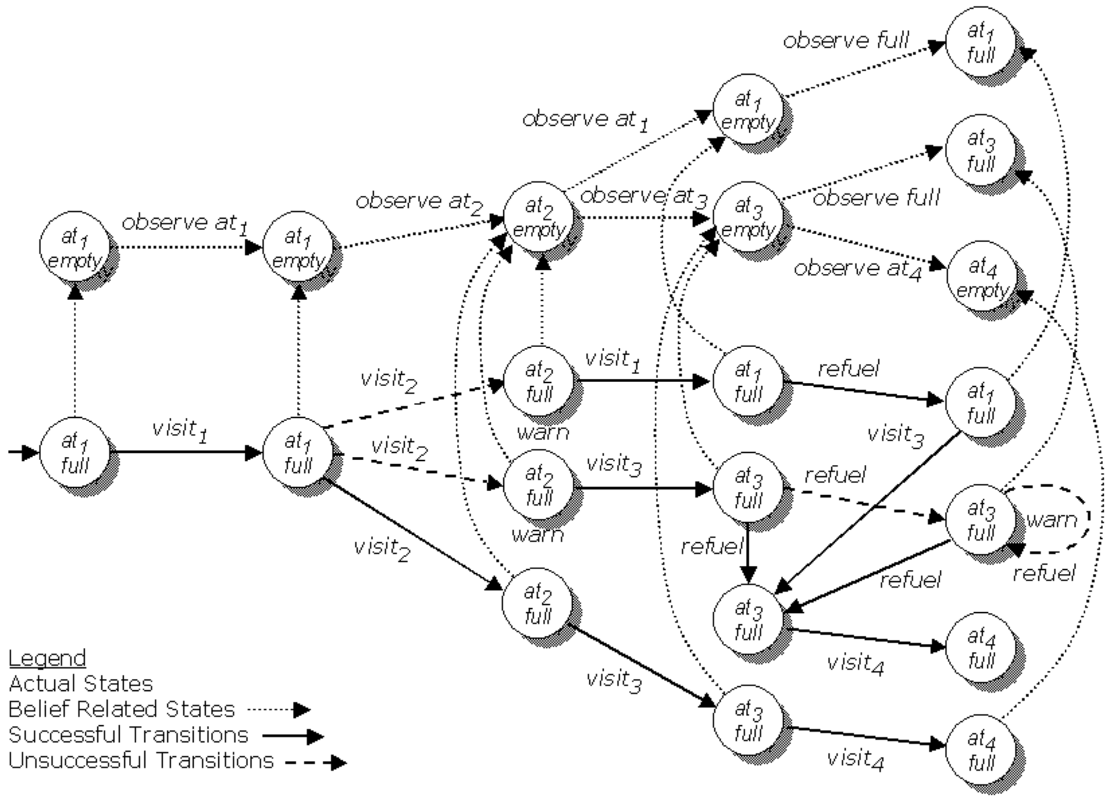

Figure 2: Agent Computation Graph for Waypoint Agent

the fuel at location 1, [*refuel*], applicable when the agent is at location 1, [$visit_1$; *refuel*] applicable otherwise, and two similar plans for the fuel at location 3).

The agent always has knowledge of its position, represented as beliefs at_i and $\neg at_j$ ($j \neq i$), and each $visit_i$ action includes a correct observation of the agent's position. The agent initially has no belief about the fuel level, but after a *refuel* action, correctly observes the state of the fuel tank, represented as a belief *full* ($= \neg empty$) or *empty*. For simplicity, the $visit_i$ and *refuel* actions always succeed, except that on occasion (here only at location 3) a *warn* event occurs. A *warn* event can therefore occur even after a refuelling action (the *refuel* action may not provide enough fuel to offset the warning).

In this example, the agent starts at location 1 (and believes this) and has a full fuel tank (though does not believe this). Thus the initial situation in the model has two $\mathcal{B}$-related alternatives (one where the tank is *full* and one where it is *empty*), but just one $\mathcal{A}$-related alternative (itself). A portion of the model for the waypoint agent is shown in Figure 2 ($\mathcal{A}$ and $\mathcal{B}$-related links between a situation and itself are omitted). An action labelling a transition corresponds to a primitive action π such that $do(\pi)$ is satisfied at the situation.

The example illustrates some of the finer points of PRS-type agent programs. First, note that the set of intentions in any configuration is indeed a set, so there can only be one instance of any given plan from the plan library at the same point in its execution in any given configuration. So, for example, repeated fuel warnings that are not acted upon have no effect on the set of intentions, since they would only be adding another instance of the same plan. However, the agent can fall into an infinite cycle when repeatedly acting on a fuel warning that is followed by another fuel warning. This is represented in the situation in the graph with a loop to itself labelled *warn*. Notice also how successfully completed intentions are dropped by the agent even when not directly acted upon, e.g. when the agent visits location 3 following a fuel warning as part of the refuelling plan, it does not have to visit location 3 again as part of the main plan, because the action $visit_3$ is removed from this plan.

In Wobcke, Chee and Ji (2005), we presented an algorithm for the construction of agent computation graphs from PRS-like agent programs. The algorithm carries out a breadth-first search of the possible environment states and configurations, beginning with a given initial state and configuration, generating $\mathcal{A}$-related alternatives and their sets of epistemic alternatives ($\mathcal{B}$-related alternatives). The basic idea of the algorithm is that, at each situation in the graph, for each possible action the agent could attempt at that state and for each possible outcome of executing that attempt (including a possible new event), the search explores a new actual situation. The algorithm keeps track of situations already visited, and terminates when no new situations are discovered (which is not guaranteed in general, since there can be infinitely many steps in iterative programs and expansions of hierarchical plans).

The main criterion for the correctness of the algorithm is the "computational grounding" condition for intention, that a situation $\sigma_{\{s,e,B,I\}}$ in an agent computation graph satisfies an ADL formula $\mathsf{I}\pi$ iff π is a future action in I. Unfortunately, this condition will only hold in restricted circumstances, since in effect, it captures our version of Bratman's strong consistency requirement for intentions and beliefs, which is built into the satisfaction conditions for intentions in BDI interpretations, but is not necessarily part of the reasoning capabilities of PRS-type agents (nor guaranteed by the designer of an agent program). The condition effectively requires that whenever the agent can reach a situation in a certain configuration, whatever action the agent chooses to attempt in that situation is possible to be executed successfully in that situation. This latter requirement is a domain-specific condition that a model checking system could be used to establish. Proving this condition may require assumptions such as the finiteness of the agent's plans and of the agent computation graph.

Conclusion and Open Questions

We believe our approach offers an intuitive formalization of a notion of intention suitable for modelling agents of a restricted class, the PRS-type agents, that is computationally grounded in that states of the agent are systematically related to their semantic counterparts. However, as previously noted, both this class of agents and the logic we have used in the modelling include simplifications, some merely for the sake of conciseness, but others inherent in the formulation. Thus a major open question is whether the approach can be applied to extensions of the PRS-like architecture, e.g. those based on alternative mechanisms for deliberation and action selection, such as scheduling. The difficulty is to redefine the semantic condition for *achieve* γ actions when the execution of a plan is interleaved with that of other plans, so as to maintain the computational grounding condition.

Further generalizations to the PRS-type architecture relate to relaxing the assumption that agents execute only one primitive action at a time. As we have recently discussed, Wobcke (2006), closer examination of some of Bratman's examples shows that the relationship between intended actions and actions performed is not one-to-one, so the reasoning of a more complex agent would need to consider complex execution strategies in relation to intentions (as in the "Video Game" example where the agent adopts a complex strategy in fulfilment of its intention), multiple primitive actions executed with the same "movement", only some of which are intended (as in the "Strategic Bomber" example, where one movement achieves both bombing the munitions plant and killing the children), and differentiating goals and postconditions of plans (as in a plan to win the lottery, in which the plan can be executed successfully even though the goal of winning the lottery is not achieved).

A further outstanding issue is the development of a theory of rationality for use with BDI agents that takes into account the uncertainty involved in decision making, an issue that receives comparatively little attention in Bratman's theory.

References

Allen, J. F. 1990. Two Views of Intention: Comments on Bratman and on Cohen and Levesque. In Cohen, P. R., Morgan, J. and Pollack, M. E., eds., *Intentions in Communication*. Cambridge, MA: MIT Press.

Bratman, M. E. 1987. *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press.

Bratman, M. E. 1990. What is Intention? In Cohen, P. R., Morgan, J. and Pollack, M. E., eds., *Intentions in Communication*. Cambridge, MA: MIT Press.

Cohen, P. R. and Levesque, H. J. 1990a. Intention is Choice with Commitment. *Artificial Intelligence* 42:213–261.

Cohen, P. R. and Levesque, H. J. 1990b. Persistence, Intention, and Commitment. In Cohen, P. R., Morgan, J. and Pollack, M. E., eds., *Intentions in Communication*. Cambridge, MA: MIT Press.

Emerson, E. A. and Clarke, E. M. 1982. Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons. *Science of Computer Programming* 2:241–266.

Georgeff, M. P. and Lansky, A. L. 1987. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, 677–682.

Harel, D. 1979. *First-Order Dynamic Logic*. Berlin: Springer-Verlag.

Holzmann, G. J. 1997. The Model Checker SPIN. *IEEE Transactions on Software Engineering* 23(5):279–295.

Pratt, V. R. 1976. Semantical Considerations on Floyd-Hoare Logic. In *Proceedings of the Seventeenth IEEE Symposium on Foundations of Computer Science*, 109–121.

Rao, A. S. and Georgeff, M. P. 1991. Modeling Rational Agents within a BDI-Architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, 473–484.

Rao, A. S. and Georgeff, M. P. 1992. An Abstract Architecture for Rational Agents. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 439–449.

Singh, M. P. 1998. Semantical Considerations on Intention Dynamics for BDI Agents. *Journal of Experimental and Theoretical Artificial Intelligence* 10:551–564.

Wobcke, W. R. 2002. Modelling PRS-Like Agents' Mental States. In Ishizuka, M. and Sattar, A., eds., *PRICAI 2002: Trends in Artificial Intelligence*. Berlin: Springer-Verlag.

Wobcke, W. R. 2006. An Analysis of Three Puzzles in the Logic of Intention. In Sattar, A. and Kang, B.-H., eds., *AI 2006: Advances in Artificial Intelligence*. Berlin: Springer-Verlag.

Wobcke, W. R., Chee, M. and Ji, K. 2005. Model Checking for PRS-Like Agents. In Zhang, S. and Jarvis, R., eds., *AI 2005: Advances in Artificial Intelligence*. Berlin: Springer-Verlag.

Wooldridge, M. J. 2000. Computationally Grounded Theories of Agency. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*, 13–22.