Learning as Abductive Deliberations

Budhitama Subagdja[1], Iyad Rahwan[2], and Liz Sonenberg[1]

[1] Department of Information Systems, University of Melbourne
[2] Institute of Informatics, The British University in Dubai, (Fellow) School of Informatics, University of Edinburgh, UK

Abstract. This paper explains an architecture for a BDI agent that can learn based on its own experience. The learning is conducted through explicit procedural knowledge or plans in a goal-directed manner. The learning is described by encoding *abductions* within the deliberation processes. With this model, the agent is capable of modifying its own plans on the run. We demonstrate that by abducing some complex structures of plan, the agent can also acquire complex structures of knowledge about its interaction with the environment.

1 Introduction

The BDI (Beliefs, Desires, Intentions) agent model [10] is a design framework commonly used in developing agents that behave both deliberatively and reactively in a complex changing environment. The main principle is to use explicit representations of the agents' own mental attitudes (in terms of attributes such as beliefs, desires, and intentions) to direct their actions and decision of choosing the appropriate predefined plan. To develop the system, the designer would define some initial mental conditions and describing some plans explicitly which correspond to the agents behavior in a repository of plans. Variability in behavior can be attained by the process of deliberation.

However, it is always possible that unforeseen conditions require some modification of the prescribed plans or knowledge instead of just alternating one plan after another. Although, the exhibition of the behavior can be adaptive in a reactive way, plans for directing or guiding the behavior in a BDI agent are still fixed in advance of the system execution. Most existing BDI frameworks are still incapable of modifying plans or *recipes* for actions at runtime.

In this paper, we present a new model of learning in BDI agents. We use *meta-level plans*, expressed in general programming constructs, to enable the agent to specify learning and deliberation steps explicitly. This enables the agent to introspectively monitor its own mental state and update its plans at runtime. The learning is regarded as a kind of deliberation process in which the agent makes plausible hypotheses about expected outcomes and creates (or modifies) plans if the hypotheses are proven. This kind of process is also known as *abduction*, a term which was coined by C.S. Peirce [6]. In this case, the agent is not just selecting the best option available but also expecting useful knowledge to be acquired if the selection fails.

This work advances the state of the art by combining the strengths of learning and BDI agent frameworks in a rich language for describing deliberation processes. In particular, our approach enables domain experts to specify learning processes and strategies explicitly, while still benefiting from procedural domain knowledge expressed in plan recipes (as opposed to generating and learning plans from scratch).

The remainder of this paper is structured as follows. the next section explain the architecture of a BDI agent. The section also describes the concept of deliberation processes as meta-level plans which can accommodate *abductions*. Section 3 then explain how learning can be described explicitly in terms of meta-level plans and deliberation processes. In that section we describe some primitives for learning and some examples of generic strategies for the experience-based plan construction. Section 4 illustrates the characteristic of the learning approach from a case study. Section 5 discusses some related works on learning intentional agents. Finally, the last section concludes the paper.

2 BDI Agent Architecture

The BDI architecture works as an interpreter interacting with different data structures. In PRS [4] as the commonly used BDI implementation model, there are four different types of data structure. Firstly, *beliefs* or *belief base* ($\mathcal{B}$) correspond to a model or knowledge about the world which can be updated directly by events captured on the sensors. Secondly, the agent's *desires* or *goals* ($\mathcal{G}$) correspond to conditions the agent wants to fulfill. The *desires* invoke finding ways to achieve them and select one (or some) to act upon. Thirdly, the selected ways to be committed for execution are the *intentions* ($\mathcal{I}$). Lastly, the knowledge of how to achieve certain desires or goals are stored in the *plans* or the *plan library*.

The common process of a BDI interpreter that drives the agent's behavior is an iteration of steps like updating beliefs based on observation in the world, deciding what intention to achieve, choosing a plan to achieve intentions, and executing the plan [12]. The interpreter goes through a control loop which consists of observation, intention filtering, and plan selection. The adopted intention is committed for execution to its end. If something goes wrong with the intention, the agent can reconsider its intention, select another plan as an alternative or just drop the intention and select another intention.

In PRS-like agents, the loop may produce a hierarchical structure of intentions. A selected intention may invoke further deliberations which produce other intentions having sub-ordinate relations with the former one. This hierarchical structure is also called the *intention structure*. The intention structure represents a stack structure consisting of goals, subgoals, and their intentions. The intention structure maintains some information about the state of the agent choices and actions, limits the number of choices to be considered at a deliberation moment, thus reducing computational complexity at every cycle. By using this structure, a goal can be broken down further to be more specific while the agent behaves reactively to changes in the environment.

2.1 Plans and Intentions

A plan represents procedural knowledge or know-how. As a knowledge for accomplishing a task, a plan would be used as a *recipe* which guides an agent in its decision making process, hence reducing search through alternative solutions [9]. In classical STRIPS planning [2], a plan consists of a set of operators or actions each with attributes like a list of *preconditions*, an *add list*, and a *delete list*.

Definition 1. *An action α is a tuple of $\langle A_\alpha, P_\alpha, \Delta_\alpha, \Sigma_\alpha \rangle$ in which A_α is the action name; P_α is a list of conditions that must be believed to be true prior to the execution of α; Δ_α is a list of conditions that must be believed to be false after the performance of α; and Σ_α are those that are believed to be true after the performance of α. Conditions are expressed as literals which can be propositions or predicate logic statements.*

A plan can be considered as an encapsulated description of actions with its consequences and contexts. It may represent just a single action or it can describe a complex relationship between actions. Similar to an action, a plan also has contextual descriptions like preconditions and effects (add or delete lists). In addition, a plan can also have attributes like a trigger (goal) and a body that describes relationships between actions.

Definition 2. *A plan π can be defined as a tuple $\langle \varphi_\pi, P_\pi, \Sigma_\pi, \Delta_\pi, B_\pi \rangle$ where P_π, Σ_π, and Δ_π are respectively the preconditions, add list, and delete list which have the same meaning as the corresponding symbols in the action definition above. The trigger φ_π is the goal that triggers the activation of the plan. The plan body B_π describes actions and their relationships.*

The plan goal states the thing that is wanted or desired by executing the plan. There are two types of goals: **achieve** and **perform**. A plan with an **achieve** goal says that a condition stated in the goal will hold or be true after performing actions described in the plan body. A **perform** goal, on the other hand, tells that actions described in the goal will be performed if the plan is executed. Actions described in a perform goal or a plan body are represented in a composite action.

Definition 3. *A composite action $\tau_\mathcal{C}[\phi_1, ..., \phi_n]$ states the relationship between actions. $\tau_\mathcal{C}$ is the type of the relation, in which $\tau_\mathcal{C} \in \Upsilon_\mathcal{C}$ and ϕ_i can be an action, a proposition, or another composite action forming a nested structure of actions relationship. If ι_α is a composite action, $\nu \leftarrow \iota_\alpha$ is an assignment of the result of the action ι_α to the variable ν.*

There can be many types of structure in $\Upsilon_\mathcal{C}$. Due to space limitations, table 1 only describes some of them which seem to be relevant and important. A variable, once bound to a value, can be used for a later purpose through variables in term parameters. For example, the composite action **seq**[**do**[$X \leftarrow$ select_object], **do**[grasp(X)]] states that an object X is selected and then grasped. The object value which is bounded by the variable X as a result of the selection action is fed into the action grasp.

In the deliberation cycle of the BDI interpreter a plan is selected from the plan library based on current goals and intentions. The selected plan is instantiated and incorporated as an intention. The intention is put on the intention structure

Table 1. Action structures and relationships

Relation type	Description
do$[\alpha]$	execute a single action α
confirm$[c]$	confirm if condition c is true in the agent's beliefs
conclude$[c]$	conclude that the condition c is true by asserting it to the agent's beliefs
wait$[c]$	wait until the condition c is true
subgoal$[\varphi]$	post the goal φ as a subgoal
seq$[\beta_1, ..., \beta_n]$	execute substructures β_1 to β_n consecutively
seq-choices$[\beta_1, ..., \beta_n]$	try to execute substructures in the list from β_1 to β_n consecutively until a successful execution of one of them
cycle$[\beta_1, ..., \beta_n, <$ **until** $c >]$	iteratively execute all substructures in the list based on the order and stop until a condition c (optional)is true

before it is executed later on. A plan instance or an intention stores an index that locates the current selected goal or action in the corresponding plan body. It also maintains information about variable bindings and states of the intention. An intention can be in a `scheduled`, `succeeds`, `fails`, `pushed`, or `wait` state.

If the plan body of a plan instance has a nested structure, then the substructure of the composite action becomes a new intention which is concatenated at the intention of that plan instance. This is also conducted for a composite action that posts a subgoal. Another plan instance for achieving the subgoal will be concatenated at that location in the intention structure.

2.2 Meta-level Plans and Abductions

In the previous section we described the model of plans and intentions in a BDI agent architecture. In this section, we explain the use of meta-level plans for controlling the deliberation. This section also shows how meta-level plans can leverage the deliberation process with abductions.

The original PRS model assumes that the deliberation process is handled by the use of meta-level plans [5]. The instance of meta-level plans can obtain information from the intention structure and change it at runtime. A meta-level plan for the deliberation process can be characterized as a plan which contains some *meta-actions* or actions that deal with goals, intentions, and plans. For example, the composite action described below shows some parts of the deliberation process.

cycle[
do[observe], $G \leftarrow$ **do**[consider_options], $I \leftarrow$ **do**[filter_options(G)],
$P \leftarrow$ **do**[select_plan(I)], **do**[intend(I, P)]]

This structure of composite actions can be put initially in the intention structure. It works as an infinite loop of **observe** for updating belief, **consider_options** for generating options, **filter_options** for selecting intentions, **select_plan** for selecting a plan instance, and **intend** that insert the selected plan to its corresponding intention and put them on the intention structure. Objects passed or

exchanged between actions are goal options (G), selected intentions (I), and a plan (P). The intention execution and reconsideration parts of the loop are skipped for simplification and it is assumed that executing the intention in the intention structure is done by the interpreter as a default process.

Based on the process of deliberation and execution, it is possible to say that the agent decides a plan and selects an action based on its beliefs and goals. We assume that if the agent has a plan for achieving a goal, it means also that the agent believes that executing the actions described in the plan will bring about the goal. This kind of process of selecting and adopting a plan instance can be regarded as a deductive inference. When there is a failure in executing the plan, a re-deliberation or re-selection process can be conducted from the beginning with a refreshed condition of beliefs. A more sophisticated technique is retaining the history of the past failures so that a failed plan instance will never be chosen for the second time.

In this paper, we suggest that *abductions* can incorporate the deliberation process so that the agent may not just re-deliberate to deal with failures and changes but also anticipate what would happen. The agent tries to develop explanations about possible failures while it tries to achieve the goal. In our model, the abduction is activated by the deliberation. The agent decides not just the goal and the intention to fulfill but also some proofs of hypothetical failures. The composite action for the deliberation cycle becomes as follows:

$$\begin{array}{l}\textbf{cycle}[\\ \quad \textbf{do}[\text{observe}], G \leftarrow [\textbf{do}[\text{consider\_options}]], I \leftarrow \textbf{do}[\text{filter\_options}(G)],\\ \quad P \leftarrow \textbf{do}[\text{select\_plan}(I)], H \leftarrow \textbf{do}[\text{make\_hypothesis}(G, I)],\\ \quad P' \leftarrow \textbf{do}[\text{select\_testing\_plan}(H, I)], \textbf{do}[\text{intend}(I, P)], \textbf{do}[\text{intend}(H,P')]]\end{array}$$

The action **make_hypothesis** generates hypotheses based on prescribed beliefs about what kind of situation would come up. The hypothesis made can be expressed as a composite action describing the sequence of events or actions that would happen. The plan for testing the hypothesis can be described as a meta-level plan. This meta-level plan involves types of actions that can capture changes in the agent's mental state (e.g. **wait**, **confirm**). The successful execution of testing will produce or revise a belief about something that is hypothesized.

When a test to prove a hypothesis fails, it will just be dropped or removed from the intention structure just like a normal plan execution. The testing plan will be re-activated in the next deliberation cycle if the same goal is still needed to be achieved. The testing plan may also update the beliefs about quality, preference, or confidence levels of a plan so that the plan will have a greater chance of being picked up by the action **select_plan** in the next cycle of the deliberation loop.

Different proving or testing strategies can be used for testing different hypotheses. For example, a hypothesis testing plan can have a composite action like the following: $\textbf{seq}[\textbf{wait}[\text{done}(I)], \textbf{confirm}[\text{success}(I)], \textbf{do}[\text{assert\_belief}(H, I)]]$. This composite action can be used to prove that the intention I will be executed successfully so that a new belief about the hypothesis H in relation with I can be asserted. The action **wait** waits for I until its finish before confirming its success.

In another case, a different testing strategy might have a more complex composite action like the following:

$$\textbf{seq-choices}[\textbf{cycle}[\textbf{wait}[\text{done}(I)], \textbf{confirm}[\textbf{not}\,\text{success}(I)], \\ H \leftarrow \textbf{do}[\text{append}(H, I)]], \textbf{do}[\text{assert\_belief}(H, I)]]$$

This composite action can be used to prove that actions in the intention I will eventually succeed or reach the goal if they are repeated for a certain number of times. The first branch of the **seq-choices** captures an unsuccessful attempt and updates the hypothesis with an additional step. The second branch is executed when a successful attempt is found and a belief about the repetition structure can be asserted. This hypothesis testing strategy still has a flaw in dealing with a single alternative of action only. An infinite loop might be produced if the goal can not be attained. However, the structure can trivially be amended by inserting some **confirm** actions on both branches to test if the length of the repetition exceeds a certain limit.

To deal with complex situations and problems, different abduction plans can be given to test different possible structures of actions upon several attempts of goal achievements. The abduction plans can be provided by the domain expert or the agent designer as heuristics for acquiring knowledge.

3 Representing Learning Processes

Learning in BDI agents can be defined as abduction-deliberation processes which can result in the improvement of the agent's performance. The approach of improvement suggested in this paper is by modifying or generating plans in the plan library. The hypotheses confirmed through the abduction process are candidate plans. The steps of confirmation are followed by a plan generation or modification. For example, the following composite action can be considered as the body of a learning plan:

$$\textbf{seq-choices}[\textbf{cycle}[\textbf{wait}[\text{done}(I)], \textbf{confirm}[\textbf{not}\,\text{success}(I)], \\ H \leftarrow \textbf{do}[\text{add\_plan\_step}(H, I)]], \textbf{do}[\text{create\_plan}(H, I)]]$$

This composite action is similar with the example of a composite action for hypothesis testing mentioned above. It captures repetitive actions for achieving the goal. However, the end result of the testing process is a new plan. In this case, the hypothesis H is a template of the possible plan. The learning plan can be said as trying to confirm that there is a sequence of repetitive actions that eventually reaches the goal. If a repetition of actions is confirmed, a new plan with a repetition or a sequence structure can be asserted.

4 Case study

This section shows some examples of learning plans and the development of the agent's knowledge when the agent is given certain tasks and is situated in a certain environment. In order to implement the experiment for studying the

characteristic of the learning agent, we have developed a special type of a BDI interpreter which supports introspective plans monitoring and modification at runtime.

4.1 The Rat's World

The Rat's World is an implemented simulation inspired by the psychological experiment of operant conditioning. An artificial rat agent is put on a designated place with some desires of getting some rewards (metaphorically some cheese). To get the reward the agent must select (press) some buttons in a particular order. Assume there are two buttons each with different colors (let say black and white buttons).

If the appropriate order has been setup so that the reward can be obtained by firstly pressing the black button followed by the white one, the rat can learn the combination by pursuing several trials of the same situation and converge to the right sequence (Some reinforcement learning algorithms like Q-learning can learn this kind of task very well). However, a simple modification can make this problem non-trivial. In particular, the situation becomes complicated when the position of the buttons is randomly swapped for every trial.

At one moment, the agent has beliefs about the buttons' positions and its own last action. Following the Markovian model of decision processes, these beliefs represent a state. Let's say, initially, the agent has the following belief based on its initial perception: `button_pos(black, white)` which states that the black button is on the left and the white one is on the right. The predicate `last_act(A)` is used to refer to the last action taken, and is added after the agent do the first action. The `A` can be `press(left)` for pressing the button on the left or `press(right)` for pressing the one on the right.

The agent is provided with some initial plans for getting the reward. One plan contains actions like **do**[`press(left)`] and the other has **do**[`press(right)`]. With these initial plans, a deliberation process will cause the agent to press the left or the right button in a random fashion to get the reward. A simple learning plan can be made which is triggered by a drop in its performance level (high rate of failures). The body of the learning plan (hypothesis testing) can be as follows

seq[
 wait[done(I)], $S \leftarrow$ **do**[observe], **seq-choices**[
  seq[**confirm**[success(I)], $P \leftarrow$ **do**[create_plan(I, S)]],
  seq[**wait**[done(I')], **confirm**[success(I')], $P \leftarrow$ **do**[create_plan(I, S),
   $G \leftarrow$ **do**[obtain_goal(I)], $P \leftarrow$ **do**[add_plan_step(P, subgoal(G))]
  ]
 ], **do**[generate_plan(P)]
] (**Learning Plan 1**)

The learning plan monitors events produced by the intention I on the intention structure. When I succeeds straightaway, a new plan is created by the action **create_plan** with the instance of actions in the intention I as its plan body and the observed state S as its precondition. Otherwise, it waits for another intention I' that succeeds straightaway, and a new plan is created with

the plan instance of the intention I, the precondition S, but a subgoal posting action is appended at the end. Learning Plan 1 produced two types of plan. One type of plan maps a belief state directly to an action. The other type maps a belief state to an action that reach an intermediate state and post the same subgoal recursively. For example, one plan produced has only the following structure in its body **do**[press(right)] and another one with the following structure **seq**[**do**[press(right)], **subgoal**[get_reward]]. Each generated plan has a precondition. For example,
button_pos(black, white) and **last_act**(**do**[press(right)]).

The experiment conducted has shown that the learning plan described above is not effective in dealing with the dynamic situation. This is because the agent can not distinguish some observed states when the buttons stay still from states where the buttons have just been swapped. The agent only relies on chances and the probability distribution of the learnt plans in dealing with uncertainties.

Figure 1(i) shows that the performance of learning with Learning Plan 1 is not much different than without learning at all. From 400 learning trials for each 20 cases, the performance on average still stays just slightly above 50% chances with a relatively high level of variability. The performance level is measured by the rate of successful attempts (getting the rewards).

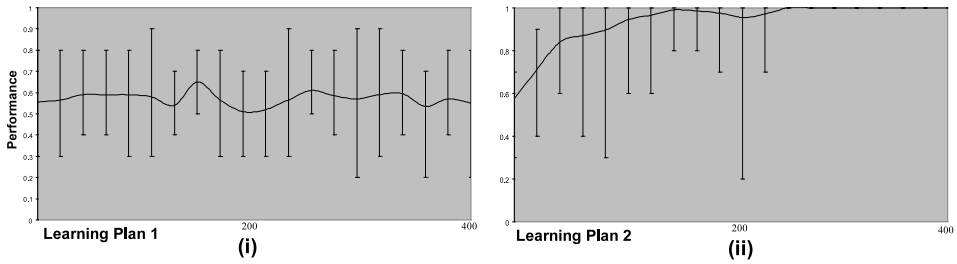

Fig. 1. Performance of agent in the Rat's World domain (i) with Learning Plan 1. (ii) with Learning Plan 2.

We modified the learning plan by encoding a different hypothesis. The agent is made to wait one more action before producing a sequence of actions. The composite action of the learning plan above is modified as follows

seq[
$S \leftarrow$ **do**[observe], **wait**[done(I)], **wait**[done(I')]
confirm[I $\neq$ I$'$], **confirm**[fails(I)], **confirm**[success(I$'$)],
$P \leftarrow$ **do**[create_plan(I, S),$P \leftarrow$ **do**[add_plan_step(P, planbody(I'))],
do[generate_plan(P)]
] (**Learning Plan 2**)

This learning plan waits for two consecutive execution of actions from intentions I and I'. If both actions are different and the first action failed while the second one succeeded, then a new plan is generated with the two consecutive

actions in the plan body and the observational state as the precondition. The result of the learning is one type of plan with a sequence of different actions as follows **seq**[**do**[press(left)], **do**[press(right)]]. The plan is also provided with appropriate preconditions based on observations. The experiment using Learning Plan 2 produced much better results. Figure 1(ii) shows that successful attempts raise the performance quickly to maximum values. In all cases, it is demonstrated that the performance always reaches the maximum. This can happen because the right interaction between the agent and the buttons can be modeled as a composite action. By hypothesizing a pattern of consecutive actions underlying the right interaction, a significant number of combinations of plan structures to be searched can be pruned.

The experiments in the Rat's World domain have clearly shown that a simple mapping between an observational state with a single action is not enough to make the agent learn the right model of interaction in a changing situation. The agent should first make assumptions about the model of its interaction with the environment. It is also indicated that applying different patterns of hypotheses influences the capability of the agent to learn something from the environment. Using complex patterns of hypotheses would also make the agent learn complex things.

5 Related Works

There is some previous work on making BDI agents learn [1, 3] by making learning as a separate process conducted by separate modules. For example, Hernandez etal. [3] apply an inductive decision-tree learning algorithm that learns new plans by feeding in logged events produced by the BDI engine to the learning program. In contrast, the learning processes suggested in this paper are mainly heuristics or knowledge that describe learning and abduction as parts of the BDI architecture. Other works have considered learning as parts of the BDI mechanism [7, 8, 11], however they still assume the creation of plans based on direct mappings of observational states to corresponding plans or actions. As demonstrated in the last section, this kind of plan creation may lead to the *perceptual aliasing* problem in which the agent would not be able to distinguish one state from another.

The advantage of putting the learning as part of the explicit knowledge in the BDI architecture is that it enables the agent developer to specify learning processes and strategies from domain or expert knowledge easily. In any case, the main feature of the BDI agent architecture is that the behavior of the agent is driven mainly by the procedural domain knowledge which can be prescribed as plan recipes as opposed to generating plans from scratch.

6 Conclusion

In this paper, we describe a model of learning in the BDI agent architecture. We use meta-level plans to program learning as an explicit part of the agent's behavior. The meta-level plans enable the agent to introspectively monitor their

own mental conditions and update their plans at runtime. The learning can be described as a process of abduction that tries to confirm the occurrences of structures of plans. The experiments conducted have shown that the agent needs to make assumptions about its possible interactions with the environment. Direct observations might be insufficient to learn the appropriate model.

Although the heuristics shown in this paper are problem specific, they can still be useful in a range of different situations. By providing a set of different types of heuristics for different classes of problems, the learning might cover various domains of application. Moreover, the learning can be multi-strategic and reactive to the change in the environment. However, further work is still needed for seeking the appropriate set of heuristics so that the approach can be practically useful.

References

1. C. F. E. Cindy Olivia, Chee-Fon Chang and A. K. Ghose. Case-based BDI agents: an effective approach for intelligent search on the world wide web. In *Proceedings of the AAAI-99 Spring Symposium on Intelligent Agents in Cyberspace Stanford University*, USA, 1999.
2. R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. In *Artificial Intelligence*, volume 2, pages 189–208. 1971.
3. A. G. Hernandez, A. E. Segrouchini, and H. Soldano. BDI multiagent learning based on first-order induction of logical decision trees. In S. O. N. Zhong, J. Liu and J. Bradshaw, editors, *Intelligent Agent Technology: Research and Development*. World Scientific, New Jersey, 2001.
4. F. Ingrand, M. Georgeff, and A. Rao. An architecture for real-time reasoning and system control. In *IEEE Expert*, volume 7(6), pages 34–44. 1992.
5. F. F. Ingrand and M. P. Georgeff. Managing deliberation and reasoning in real-time AI systems. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning*, San Diego, California, 1990.
6. L. Magnani. *Abduction, Reason, and Science: Processes of Discovery and Explanation*. Kluwer Academic/Plenum Publishers, New York, 2001.
7. E. Norling. Learning to notice: Adaptive models of human operators. In *Second International Workshop on Learning Agents*, Montreal, 2001.
8. E. Norling. Folk psychology for human modelling: Extending the BDI paradigm. In *Proc. of the Third Int. Joint Conf. on Autonomous Agents and Multi Agent Systems (AAMAS-04)*, New York, NY, July 2004.
9. A. S. Rao. A unified view of plans as recipes. In *Contemporary Action Theory*. Kluwer Academic, Netherlands, 1997.
10. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. San Francisco, 1995.
11. C. Sioutis and N. Ichalkaranje. Cognitive hybrid reasoning intelligence agent system. In *Knowledge-Based Intelligent Information and Engineering Systems, 9th International Conference, KES 2005, Melbourne, Australia*, volume 3682 of *LNAI*, pages 838–843. Springer, 2005.
12. M. Wooldridge. *Reasoning about Rational Agents*. MIT Press, Cambridge, 2000.