12-2006

Plans as products of learning

Samin KARIM

Budhitama SUBAGDJA
Singapore Management University, budhitamas@smu.edu.sg

Liz SONENBERG

Plans as Products of Learning

Samin Karim
Department of Information Systems
University of Melbourne
Melbourne, Australia
mkarim@pgrad.dis.unimelb.edu.au

Budhitama Subagdja
Department of Information Systems
University of Melbourne
Melbourne, Australia
subagdja@pgrad.dis.unimelb.edu.au

Liz Sonenberg
Department of Information Systems
University of Melbourne
Melbourne, Australia
l.sonenberg@unimelb.edu.au

***Abstract*— This paper presents motivations and current related work in the field of *plan learning*. Additionally, two approaches that achieve plan learning are presented. The two presented approaches are centred on the BDI framework of agency and have particular focus on plans, which, alongside *goals*, are the means to fulfil intentions in most pragmatic and theoretical realisations of the BDI framework. The first approach is a hybrid architecture that combines a BDI plan extractor and executor with a generic low-level learner. The second approach uses hypotheses to suggest incremental refinements of a priori plans. Both approaches achieve plan generation that is a result of experiential learning. We conclude by discussing issues related to these two approaches, and from other related work.**

I. Introduction

Planning has been a research problem tackled by a vast number of agent researchers. Learning has also received considerable attention in the agent community. The domain of such research has primarily focussed on situated agents, such as mobile robotics. Planning and learning are intertwined processes in an agent's reasoning engine [1]. However, in many cases, the result of learning is to produce singular actions at every search episode, which necessitates the engine to repeat the cycle of sense-select action-execute-learn at every discrete step. There has been little work done on the tighter fusion of learning and planning.

Learning has traditionally been implemented at the agent's lowest level of abstraction. Many such learning paradigms and frameworks, such as reinforcement learning and connectionist networks (such as neural networks) learn knowledge at a fine grained level [2] that sometimes precludes semantic clarity that, for instance, allows more intuitive human understanding of the acquired knowledge [3]. Indeed, the information acquired by such techniques can scarcely be labelled 'knowledge', but more appropriately as 'data'.

It has been the focus of some researchers to achieve learning at a higher level of abstraction (see section III). The motivations for striving for this goal go beyond attaining a higher degree of human understanding of the data. Knowledge acquired at a high level of abstraction can be more flexible and general [3], and in some cases offer performance advantages [4], [5], [6]. Indeed, we as humans like to think in terms of concepts that are familiar to us; terms that are, perhaps, symbolic in nature and are consistent with our cognitive patterns of thought. In designing and analysing an artificial intelligent agent, we may feel confident in its ability if the knowledge it possesses is in a familiar representation, and not in terms of neural network weights and arbitrary states and actions, per se. Section II goes into more detail on these issues.

One instance of an abstract representation are *plans*. Based on the notions of *BDI agency*, plans fulfil *intentions* and are grouped in terms of specifically assigned *goals*. All processes are based on an agent's internal *beliefs*. The Belief-Desire-Intention (BDI) framework [7], [8], [9] is both a folk psychologically grounded model of agent reasoning and a practical framework for defining agent reasoning artifacts in terms of the mental states: beliefs, desires and intentions. BDI agents pursue given goals (desires), adopting and committing to appropriate plans (intentions) according to its current set of data about the state of the world (beliefs). In this way agents behave as folk-psychologically plausible caricatures of humans with these mental attributes. Our execution model for BDI agents follows the standard model adopted by many, such as in the procedural reasoning system (PRS) model of Ingrand et. al [10]. The cycle first searches for an appropriate goal to fulfil depending on its beliefs about the world (state) and the set of a priori defined goals. The BDI agent then proceeds to find plans, which are execution recipes, to fulfil the intention to achieve this goal in the given context. The cycle iterates until all goals are met, or some other termination condition arises.

We believe that learning in terms of these BDI concepts (with plans as the central representation) is a plausible means to achieve learning at a high level of abstraction, and hence, we focus on the plan representation as the basis of our hypothesis. The primary aim of this paper is to provide motivations for using plans as a representation for knowledge acquired through learning. Sections II and III will outline claims and related work that argue this case, respectively. We then present two architectures as example plan learning implementations that further support our hypothesis. The first architecture (section IV) employs a hybrid technique in which a low-level learner is combined with a high-level BDI-based knowledge extractor and executor. The second approach (section V) uses a hypothesis generator to amend existing BDI-plans by way of suggesting and executing plans and updating intentions accordingly. Learning is achieved using meta-level plans that monitor the execution of plans. We discuss the ideas and claims of the paper in section VI, and conclude in section VII.

II. Plans as a form of Knowledge in Learning

We now focus on some claims that we put forward in order to support our hypothesis that plans are a useful and effective knowledge representation for learning. There are performance advantages to using plans as opposed to representations of lower abstraction, such as rules and connectionist (eg. neural) networks. Plans also offer policy compression when compared to a rule-based system [6]. Section III covers related work that supports the above claims, and experiments and discussions of the two presented architectures, detailed in sections IV and V, also goes towards supporting these claims.

Plans (with their associated goals) are a rich representation that encapsulates *context* in a temporally extended manner. Single actions do not have this distinction. In some domains, the context of actions are important considerations during the learning process. For instance, when compared to a rule-based system, two rules that are learned in sequence may be in conflict (ie. they may be 'cancelling' the agents' efforts to achieve a common goal), but if the actions are appended to a plan, any conflicts that arise between two or more actions are handled via the penalisation, or removal, of that particular plan from the knowledge base. Section V-B describes results from experiments that supports this claim.

There can be a counter argument that plans have a level of abstraction that may overlook some fine-grained level of granularity of changes in the environment. This condition would point out the drawback of the plan representation. However, recent studies about heuristics [11] have shown that under certain conditions, a lack of knowledge can be beneficial for an agent to make inferences. In fact, one experiment described in section V-B demonstrates that ignoring a fine-grained level of abstraction in the contents of some observational states during the learning process can avoid conflicting information. It means also that under certain circumstances, a fine-grained level of abstraction may not be suitable for learning.

At a qualitative level, it can be argued that the BDI representation, specifically plans and goals, are a more *intuitive* representation of knowledge than other representations such as rules. This intuitiveness may make the learnt knowledge base of the agent more amenable to analysis by humans. There is work that, either directly or indirectly, supports this notion (eg. [10], [12]). The field of psychosemantics and those who propose that there is a natural language of thought have posited that, indeed, there is a structured, semantic language by which humans think. Fodor [13], [14] claims that humans think in a semantic structure represented as mental states of beliefs, desires and intentions, which the BDI representation is based on.

The knowledge required by the agent in a *complex* domain might be too difficult and/or time consuming to hand-code. It may be unreasonable and impractical for domain experts to define the knowledge base, where in the case of BDI-based systems they would be required to write plans with assigned context and goal information. In such a case, a learning capability would significantly reduce the domain expert's involvement.

III. Related Work

It has been recognised that the learning process requires more than just simple rule mappings, or collections of connectionist networks with arbitrary weights and connections, as products of learning. Some works have been done in automating the extraction of plans from an agent's experiences. Lebiere and Wallach [15] introduce learning in the ACT-R framework [16] in a hybrid configuration. By using simple recurrent networks (SRNs) at the implicit, bottom level, subsequent elements of a plan are based on the current input sequence and temporal (historic) context. The chunk mechanism in ACT-R is used to represent goal stacks, where one goal is active at any one time. Rules ($condition \rightarrow action$ profiles) activated by the bottom-level are implicitly encoded in the goal chunk such that they form a sequence (plan). These implicit encodings are fragments of the plan. This binding between implicit and explicit representational levels is reminiscent of Sun's CLARION architecture [5], and a similar chunking mechanism is used to formulate plans in the Soar architecture [17]. Sun also worked on plan acquisition [6] in CLARION in a similar vein to Lebiere and Wallach's work with ACT-R. However, in Sun's attempt the plans are extracted/generated offline by performing a beam search over the acquired rules in the explicit layer.

Beyond hybrid mechanisms, some work has been done as well in automating creations of partial plans based on symbolic reasoning. Malec and Nowaczyk [18] have recently developed an architecture that formulates partial plans that are learned and adapted using inductive logic programming (ILP), and consists of three components. A *deductor* performs deductive reasoning and primarily to generate partial plans. The *actor* executes and monitors actions. A *learning component* integrates the outputs of both deductor and actor to change parameters of subsequent executions. However, again, this learning is achieved offline. A similar approach has been used with the BDI architecture in a multi-agent setting. The learning is conducted symbolically using inductive decision-tree learning [19]. New partial-ordered plans can be generated by generalising statements from historical log of activities using inductions.

Some architectures are also built to extract plans while performing some online tasks. A classic model in plan generations through the process of learning is the Case-Based Planning [20]. Using a Case-Based Reasoning system, plans can be created or revised using cases from past experiences. The Case-Based Reasoning approach to modify plans has also been extended to incorporate the BDI agents model [21]. The processes of deliberation, execution and learning are all conducted under the mechanism of a Case-Based Reasoning system. Plans are also selected and executed based on degrees of similarity between cases. Garland and Alterman [22] present an architecture in which coordination plans are generated around a priori defined coordination points. These plans are extracted from execution traces that are stored as cases in a case base. The architecture is designed to cut computation

time by selecting and executing these traced plans in order to alleviate the need for first-principles planning.

Another mechanism for modifying plans makes use of the plans' inherent structure. Beetz has applied a plan-modification mechanism for a BDI-like architecture in a robotic domain using the reactive plan language (RPL) [23]. The plans are modified by a particular module that changes the plan directly as parts of the anticipation process to avoid execution failures. Existing plans are projected into the future through a simulation process to identify possible failures. Corresponding plans are modified or new ones are created in order to anticipate the execution failures.

These works on generating plans based on experiences or run-time executions have led us to explore the characteristics of plans as the result of learning in the autonomous agent domain. In this paper, we look at some beneficial aspects in producing plans by learning using the BDI agent architecture, and present two architectures that achieve plan learning.

IV. Learning Plans: A Hybrid Approach

A. The Architecture

In this architecture, first presented in [24], a hybrid approach is taken in which a top level interacts with a bottom level in order to extract knowledge represented at a higher level of abstraction in the form of BDI-styled plans. The top level, which we call the *plan generation subsystem* (PGS), is based on the BDI framework and monitors low-level executions at the bottom level. The bottom level can be any module that performs atomic (primitive) actions, usually based on a representation of low abstraction such as rule-based systems consisting of $condition \rightarrow action$ profiles, or neural networks with knowledge stored as neural weights and connections. In [24] the bottom level is a reinforcement learning module called FALCON that learns state-action pairs (rules).

The essential heuristic of the architecture is described as follows. PGS generates plans by utilising a priori data in the form of *clues* that are in turn associated to specific goals. These goal-clue tuples are defined by a domain expert at design/configuration time, and are a central component within the PGS architecture. Clues can be thought of as *triggers* for recording plans that are extracted from actions executed in sequence by the bottom level. The execution of actions from within plans and from recommendations from the bottom level are chosen via a switching mechanism during runtime based on appropriate utility measures provided by the bottom-level and mirrored at the top level. The detailed heuristic is given in Algorithm 1. An example PGS configuration consisting of clues, and corresponding bottom-level and top-level modules is given in the next subsection (IV-B) and serves to illustrate this architecture.

Algorithm 1 The PGS plan generation and hybrid system integration heuristic

Require: $n > threshold_1$. $\mathcal{C} = \{C_1, C_2, ..., C_n\}$ a set of bottom level state-action tuples, where $C_i = (s_i, a_i)$ in which s_i is the state and a_i the action. Initialise plan library, P, and set selected plan, $p_s = null$

1: **for** each consecutive (s_i, a_i) that occurs in the execution cycle **do**
2: $p_s \leftarrow$ EXECUTINGPLAN()
3: **if** $p_s \neq null$ (ie. p_s selected) **then**
4: $r \leftarrow$ EXECUTEACTION(NEXTSTEP(p_s))
5: $P \leftarrow$ REINFORCEPLAN(p_s, r, P)
6: **continue loop**
7: **end if**
8: $\mathcal{G} \leftarrow$ DETERMINEACTIVEGOALS(s_i)
9: **if** $\mathcal{G} = \varnothing$ **then**
10: EXECUTEACTION(a_i)
11: **continue loop**
12: **else**
13: $G \leftarrow$ CHOOSERANDOMGOAL($\mathcal{G}$)
14: $p_s \leftarrow$ SELECTPLAN(G, P)
15: **end if**
16: **if** $p_s \neq null$ (ie. p_s selected) **then**
17: $r \leftarrow$ EXECUTEACTION(NEXTSTEP(p_s))
18: $P \leftarrow$ REINFORCEPLAN(p_s, r, P)
19: **continue loop**
20: **else**
21: $p_e \leftarrow$ FINDPLANTOAPPEND(s_i, a_i)
22: **if** $p_e = null$ (ie. p_e not found) **then**
23: APPEND(p_e, a_i)
24: **else**
25: $p_n \leftarrow$ CREATENEWPLAN(s_i, a_i)
26: $P \leftarrow$ ADDPLANTOLIBRARY(p_n, P)
27: **end if**
28: EXECUTEACTION(a_i)
29: **continue loop**
30: **end if**
31: **end for**

B. Experiments

The PGS architecture has previously been tested by conducting experiments in a grid-world *minefield navigation* domain [24]. In this paper a more complex predator-prey (or *pursuit*) domain has been used as the test bed. In this domain, there are four predator agents and one prey agent in a non-toroidal grid. Each agent can move up, down, left or right. The task of the predators is to surround the prey on its north, east, south and west sides, and the learning is implemented exclusively for predators in these experiments. The prey possesses fixed, non-adaptive behaviour, which is to move away from adjacent predators and move towards the grid edge. As a result, the prey is limited to perception of adjacent agents and bearing of walls. If the prey manages to reach the grid boundary, or the number of cycle steps exceeds a defined threshold, or if the predators move in each others' occupied squares (ie. a 'conflict') then the prey wins. If the predators surround (capture) the prey, the predators win. The reward functions for each agent type reflect these rules.

The learning engine that learned the predators' control strategy was firstly implemented using FALCON, an RL based

learning algorithm [25]. The data at this bottom-level is represented as rules, ie. $condition \rightarrow action$ profiles, where the condition would simply be the situation as perceived by the predator agent. In these experiments, the situation as perceived by each predator agent is a vector of the (numerical) degree of *exposure* of the prey, and the prey's bearing with respect to the predator. Exposure is a simple measure of how exposed a prey is. For example, if the prey is captured (ie. four predators surrounding the prey) then the prey's exposure is at the absolute minimum. If the predators are further way from the prey, the prey has more room to move and the exposure is larger.

We augment the initial implementation by adding our PGS hybrid system (based on Algorithm 1), where PGS is the top-level, and the FALCON learning module is the bottom-level (similar to the configuration in [24]). The domain expert defined goal-clue tuples defined in this instance are as follows:

1) **Goal:** Minimise exposure.
   Clue: Consider plans when the exposure of the prey is reducing from the previous step (meaning the predators are increasing their "coverage" of the prey).
2) **Goal:** Minimise distance.
   Clue: Consider plans when the distance between predators and prey is reducing from the previous step.

The above clues are utilised as follows. In Algorithm 1, when the DETERMINEACTIVEGOALS(s_i) function is called, these clues serve as a kind of *mask* that essentially determines if the current situation, s_i, is relevant to the associated goal. If the clue condition is true for situation s_i, then the DETERMINEACTIVEGOALS(s_i) function will append the goal of the goal-clue tuple to the set of active goals $\mathcal{G}$ returned by the function. To elaborate on goal-clue tuple 1 above, for example, if at situation s_i the exposure of the prey is reducing from the previous step, then PGS will start recording the resulting actions into a plan, and terminate the plan once the clue condition no longer holds. This plan is then stored in the plan library and recalled for execution and reinforcement at a later stage if it is deemed relevant to another future situation.

Given the plan library, $\mathcal{P}$ and individual plans, p_i, such that $p_i \in \mathcal{P}$, and

$$p_i =< G_i, C_i, \mathcal{E}_i >$$

where $\mathcal{E}_i$ is the set of steps (elements) in plan p_i, G_i is the plan's goal, and C_i is its context. Context in this experiment domain is simply the position of predators and prey in the grid (specified as coordinates), and the goal is one of the selected goals in the set of goal-clue tuples listed above. G_i and C_i are formed during plan conception. In these experiments, plans consist of actions from all four predators and are combined in a single shared plan. Hence, one shared plan contains the plans for all four predator agents. The plan elements, e_j, are defined thus:

$$e_{i,j} =< a^1_{i,j}, a^2_{i,j}, a^3_{i,j}, a^4_{i,j} >$$

where $e_{i,j}$ is the jth element in the set of plan elements $\mathcal{E}_i$ ($e_{i,j} \in \mathcal{E}_i$) and $a^n_{i,j}$ is the action of predator n of the jth element, e_j of plan p_i. Plan elements, $e_{i,j}$, are appended as steps to p_i during subsequent cycles. During execution, a plan, p_i is selected such that $s_i \rightarrow C_i$ and $G_i \in \mathcal{G}$, where $\mathcal{G}$ is the set of active goals for s_i (see Algorithm 1, steps 8, 13 and 14).

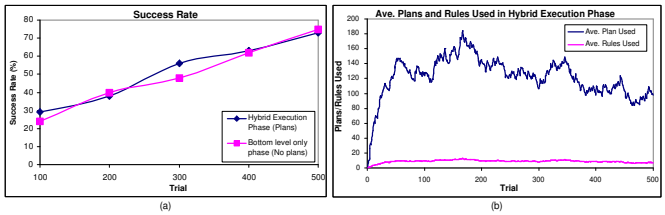

Fig. 1. PGS results (a) Success Rate after 100 trial increments, (b) Running average of plans used compared to rules used during the PGS hybrid execution phase

The experiment was set up as follows. In each trial, the predators and prey are positioned randomly in a 16×16 grid. The predators and prey have 200 time steps (cycles) in which to achieve their respective goals, with the rules of the game as described above. The experiment run consists of 1000 consecutive trials. The first 500 trials were run with only the FALCON learning engine activated (ie. the PGS plan generation and execution heuristic inactive). After these initial 500 trials, the PGS module was activated and began generating and executing plans for the next 500 trials (ie. trial 500 to 1000). This arrangement ensures that the knowledge base of the FALCON module is sufficiently primed (after 500 trials) so that the PGS module can effectively acquire plans from the bottom-level FALCON module.

Figure 1(a) shows the performance (success rate) of each configuration over 500 trials, at 100 trial intervals. The "Hybrid Execution Phase" data line represents the success rate of the hybrid system (ie. plan generation and execution activated), which matches very closely to that of the FALCON-only execution phase (ie. only the bottom-level FALCON module activated), represented by the "Bottom-level only phase" data line. Figure 1(b) shows how often plans (from the PGS module) are executed compared to rules from the bottom-level FALCON module during the hybrid execution phase. The graph shows that plans are used overwhelmingly more than rules. As the success rate increases at the later stages (shown by the upward trend of Figure 1(a)), the number of steps required for the predators to capture the prey reduces, as evidenced by the slight reduction in plans (and rules) used towards the end of the experiment run (Figure 1(b)), indicating that efficiency of the hybrid execution is improving with every trial.

V. LEARNING PLANS: A HEURISTIC APPROACH

A. The Architecture

The last section described an approach of learning that generates plans in a multi-level hybrid system using a module that monitors the execution in one level and create plans in another level. In this section, another approach is explained. Instead of separating the learning process from the main

performance execution, we use a model of intentional learning in which the processes of learning are described explicitly in terms of plan representation stored in a plan library. Using the BDI agent architecture, learning becomes part of the agent's deliberation and execution process.

The model of intentional learning which is applied in the BDI agent architecture is described in more detail in [26]. This section only presents some main ideas of the model. The presentation focuses on how the intrinsic structure of the plan representation can express different types of heuristics for acquiring procedural knowledge.

We suggest that the deliberation processes in the BDI agent can also be controlled by some meta-plans. A meta-plan is a plan which contains some *meta-actions* or actions that monitor or update the internal states of the agent such as beliefs, goals, intentions, and plans. It is suggested in [26] that the deliberation process can be incorporated with *abductions* so that the agent may also anticipate what would happen. The process of weighing and selecting alternatives can be enriched with hypothesis making and testing. In our model, the abduction is activated by the deliberation. Normally, if there is a failure or an unexpected event, the agent needs to re-deliberate to decide which plan should be adopted next regarding the current failure. We propose that during the re-deliberation, the agent also anticipates possible failures by making a hypothesis and testing it on the run. The abduction process can be realised by a meta-plan that monitors and tests the outcomes of another plan execution based on a certain hypothesis.

An example of the application of an abduction meta-plan is that when the execution of an intention i fails because its goal is not achieved, a meta-plan for abduction is activated which monitors the outcomes of the next execution of i or the achievement of the goal associated with i. The information of the status of a particular intention and the cause that made that status can be obtained from the intention structure. Let $< i >_{fails}$ and $< i >_{success}$ denotes the condition above in which i fails as the goal is not achieved and the condition in which i succeeds. An abduction plan can have a plan body as follows:

$$\textbf{seq}[\circlearrowleft [i]_{done}; ? < i >_{fails}; \circlearrowleft [i]_{done}; ? < i >_{success}]$$

which refers to a sequence of: waiting until the execution of i is done, and checking the status of i if it is fails because of the unachieved goal. The next steps of the sequence consists of, again, waiting for i to finish, and then a confirmation of its success. $\circlearrowleft$ and ? symbols denote wait and confirm action respectively, which is explained in detail in [26].

The abduction plan above can be used to test if the two consecutive executions of i would achieve the goal. The hypothesis is proven if the abduction plan is successfully executed. Consequently, the hypothesis can be considered to be failed and rejected when the abduction fails. The abduction plan can be extended to become a learning plan by incorporating certain types of meta-actions which modify or add plans in the plan library of the BDI agent based on the hypothesis to be confirmed. A successful attempt of abduction is followed by a corresponding plan generation.

To deal with different situations and problems, different learning plans can be given with different abductions to test different possible structures of actions upon several attempts of goal achievements. The idea is that a learning plan is provided by the domain expert or the agent designer as a heuristic for acquiring procedural knowledge. Although a prescribed heuristic might only work effectively in a specific environment and condition, the use of plan library as a repository of different learning heuristics may be in line with the adaptive toolbox approach [27]: by giving a bundle of middle-range heuristics for different classes of situations, the agent can adapt its behaviour to different situations and changes.

B. Experiments

This subsection explains a simple experiment conducted to test the idea of the approach and a brief analysis of the results in relation with the model. We have developed a special type of a BDI interpreter which supports the realisation of meta-plans for abductions and plan modifications at runtime. The domain testbed is different from the one used in the hybrid approach. We use the Rat's world domain.

The Rat's World is a domain of the simulation inspired by the psychological experiment of operant conditioning as a testbed for exploring the properties of learning plans. 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 colours (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. Another difference with typical reinforcement learning tasks is that the learning activities are conducted on-line in a continuous manner instead of off-line learning with discrete learning episodes.

We applied two initial plans, each consisting of a single button pressing action. Instead of pressing the button based on its colour, primitive actions are defined as pressures based on locations (press left and press right). From these initial plans, the default deliberation and execution processes will produce a series of random attempts of buttons pressing. We have applied two different types of learning plans so that some characteristics of learning based on the kind of composite structures of plans can be evaluated.

The first learning plan (Learning Plan 1) monitors a successful attempt of an intention to reach a particular goal. It generates plans each consists of some observed states as

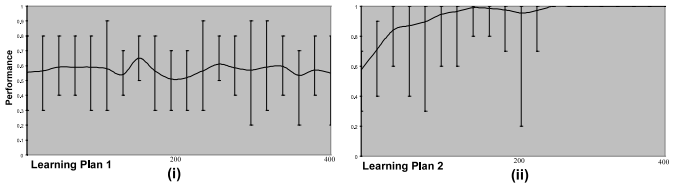

Fig. 2. Performance of agent in the Rat's World domain (a) with Learning Plan 1. (b) with Learning Plan 2.

the preconditions and a single primitive action in the plan body. Another type of plan may also be generated with an additional action that activates a subgoal. Learning Plan 1 is a plan-based BDI version of rule learning. The second learning plan (Learning Plan 2) monitors two different consecutive intentions that successfully reach the goal at the end. The plan generated consists of some observed states just before the first attempt and the plan body becomes a sequence of two different intentions that have been monitored. For the reason of space the detail learning plans and their outputs are not shown in this paper. Interested readers can refer to [26] for more comprehensive explanation.

The experiment conducted has shown that Learning Plan 1 is not effective in dealing with the dynamic situation. Figure 2(a) shows that the performance on average still stays just slightly above 50% chances with a relatively high level of variability. In fact, the performance does not differ much from random behaviour (ie. no learning at all). The performance level is measured by the rate of successful attempts (getting the rewards). On the other hand, Learning Plan 2 produces much better results. Figure 2(b) shows that successful attempts raise the performance quickly to maximum values. In all cases, it is demonstrated that the performance always reaches the maximum.

The experiments in the Rat's World domain have clearly indicated that simple mappings between observational states with single actions is not enough to make the agent learn the right model for achieving the goal in a changing situation. This is because the agent can not distinguish some states when the buttons stay still from states where the buttons have just been swapped. The learning plan has just enforced the creation of another new plan despite the inconsistency. The simple rule mapping strategy only relies on chances and the probability distribution of the learnt plans in dealing with uncertainties. On the other hand, by the Learning Plan 2, the generated plan has a composite structure that coincidentally fits with the instance of the appropriate interaction between the agent and the button. An ignorance of possible action failures can be advantageous when the right interaction can only be modelled as a plan structure.

VI. Discussion

Both of the architectures presented in this paper have shown that plans can be learned within the BDI framework and that their execution is comparable or superior in performance to the rule-based solution. The experiments involving the hybrid architecture approach (section IV) described in section IV-B has shown that the performance of plans match that of the rule-based bottom-level. The results of the heuristic based approach (section V), in section V-B, has also shown that the performance of the plan-based approach outperforms the rule-based approach.

The results from the hybrid approach experiments show that plans can be effectively extracted from generic low-level learning modules. The architecture involves a loose coupling between the high-level plan generation subsystem (PGS) to the bottom-level learner (which in the experiments was the FALCON RL engine). The interaction between the two levels are through the actions recommended and situations experienced at the bottom-level. The actual mechanism of learning, or quite simply action execution, at the bottom level, is not considered by the top-level PGS module. As a result, the bottom-level can be almost any type of action execution module, with or without a learning capability. This architecture can be used with a diverse range of bottom-level module types. The initial empirical investigation involves the use of an RL bottom-level (FALCON), which is one instance of the type of bottom-level that can be utilised. Other learning modules (eg. neural networks) can be used at the bottom level – the only requirement is that an action and situational context must be visible to the top level for effective plan generation to take place. Indeed, other categories of bottom-levels can potentially be used. For instance, a *human* could be the engine behind action recommendations at the bottom-level, and the hybrid system would then function as a human plan recognition system. Most hybrid architectures, regardless of the degree of coupling between interacting levels, do not possess this quality [5], [28]. Hence, plan generation is more the result of *observations* of another separate action executor, such as (possibly) a human or an RL learning module.

The experiments based on the heuristic based approach have demonstrated a characteristic of actions and the context in which they are executed. Under certain circumstances the agent needs to make assumptions about its possible interactions with the environment. Direct observations might be insufficient to learn the appropriate model. The use of assumptions about composite actions structure and the consideration of learning as parts of intentional actions has often been ignored in the machine learning domain. Most machine learning algorithms assume the separation of the learning phase from actual execution (which is when experience is usually gathered). This assumption contradicts the notion of experiences as a source of improvement and likewise neglects the structure of the environment the agent resides in. It is often assumed that the computer program or the agent already has complete, prescribed preferences over belief states and actions. The agent needs to adjust the relations between pre-set cues and corresponding actions which were also prescribed. However, in many application domains it is the cues themselves that must be searched for. Furthermore, some situations require the agent to plan or coordinate a number of actions rather than just to select a single action step.

A common theme between the two approaches presented in this paper is that in order to realise plan learning within the BDI framework, some domain expert input has been required. In the hybrid based approach, this took the form of specific goal-clue tuples, that essentially are cues that trigger the recording of plans with specifically assigned goals and context conditions. The heuristic based approach has taken a similar strategy, however, the specification from the domain expert is taken in the form of meta-plans that drive the learning process, and have the same form as the plans that are generated. The meta-plans are considered as middle-range heuristics that deal with different learning conditions.

VII. Conclusions and Future Work

We have discussed claims and related work on the hypothesis that plans are a suitable representation for learned knowledge. Two different architectures that achieve learning in terms of plans have been presented in the paper. The first architecture (section IV) employs a hybrid approach, which combines a plan generation module (PGS) to a generic bottom-level module that executes actions that are (possibly) learned from experience. The second approach (section V) involves the use of hypotheses that amend plans during execution. In both cases, expert input is required to drive the learning process.

One major hypothesis is that learning in terms of plans may attain performance advantages, which we have supported with our experiments and discussions of related work. In particular, we have shown through experiments (section V-B) that cases where the context of actions (with respect to other actions executed within a plan), rather than actions considered in isolation, are important considerations in the learning process. The plan representation (alongside other associated BDI artifacts such as goals) is also argued to be a more intuitive representation for humans to understand, and hence makes the resulting knowledge base amenable to analysis by human operators of the system. This is an important consideration in engineering an adaptive system. Traceability and reliability are critical factors, such as in some real world, mission critical domains.

The experiments on both heuristic-based and hybrid approaches have shown that by learning the plans structure, the rate of the performance improvement is comparable or sometimes even better than the rate in standard learning mechanisms like reinforcement learning. However, further work is still needed for seeking the appropriate plan management and learning heuristics so that the approaches can be used practically. Further analysis is also needed to associate the problem of plan learning with different levels of abstraction to find the right granularity of the learnt plan representation.

References

[1] J. Bryson, "Cross-paradigm analysis of autonomous agent architecture," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 12, no. 2, pp. 165–190, 2000.

[2] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[3] M. Minsky, "Logical versus analogical or symbolic versus connectionist or neat versus scruffy," *AI Magazine*, vol. 12, no. 2, pp. 34–51, 1991.

[4] A. McGovern and R. S. Sutton, "Macro-actions in reinforcement learning: An empirical analysis," University of Massachusetts, Department of Computer Science, Tech. Rep. 98-70, 1998.

[5] R. Sun, *Duality of the mind: A bottom-up approach toward cognition*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc., 2002.

[6] R. Sun and C. Sessions, "Learning plans without a priori knowledge," *Adaptive Behaviour*, vol. 8, no. 3/4, pp. 225–254, 2000.

[7] M. E. Bratman, *Intention, plans, and practical reason*. Harvard University Press,, 1987.

[8] M. E. Bratman, D. Israel, and M. Pollack, "Plans and Resource-Bounded Practical Reasoning," in *Philosophy and AI: Essays at the Interface*, R. Cummins and J. L. Pollock, Eds. Cambridge, Massachusetts: The MIT Press, 1991, pp. 1–22.

[9] A. S. Rao and M. P. Georgeff, "BDI-agents: from theory to practice," in *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco, 1995.

[10] F. F. Ingrand, M. P. Georgeff, and A. S. Rao, "An architecture for real-time reasoning and system control," *IEEE Expert*, vol. 7, no. 6, 1992.

[11] G. Gigerenzer, P. M. Todd, and the ABC Research Group, *Simple Heuristics That Make Us Smart*. New York: Oxford University Press, 1999.

[12] E. Norling, "Learning to notice: Adaptive models of human operators," in *Proceedings of The Second International Workshop on Learning Agents*, Montreal, Canada, 2001.

[13] J. A. Fodor, *Psychosemantics*. Bradford Books/MIT Press, 1987, ch. Why there still has to be a language of thought, pp. 135–167.

[14] W. G. Lycan, Ed., *Mind and Cognition: A Reader*. Blackwell Publishers Ltd., 1998, ch. 13: Psychosemantics, pp. 312–337.

[15] C. Lebiere and D. Wallach, "Sequence Learning in the ACT-R Cognitive Architecture: Empirical Analysis of a Hybrid Model," in *Sequence Learning: Paradigms, Algorithms and Applications*, R. Sun and C. L. Giles, Eds. Springer, Jan. 2001, vol. 1828, pp. 188–.

[16] J. R. Anderson, *Cognitive Psychology and Its Implications*. Worth Publishers, 2000.

[17] A. Newell, *Unified Theories of Cognition*. Cambridge, Massachusetts: Harvard University Press, 1990.

[18] J. Malec and S. Nowaczyk, "Deduction and Exploratory Assessment of Partial Plans," in *IJCAI Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains*, V. Bulitko and S. Koenig, Eds., Edinburgh, UK, August 2005.

[19] A. G. Hernandez, A. E. Segrouchini, and H. Soldano, "BDI multiagent learning based on first-order induction of logical decision trees," in *Intelligent Agent Technology: Research and Development*, N. Zhong, J. Liu, S. Oshuga, and J. Bradshaw, Eds. New Jersey: World Scientific, 2001.

[20] K. J. Hammond, *Case-Based Planning: Viewing Planning as a Memory Task*. Boston: Academic Press, 1989.

[21] C. Olivia, C.-F. Chang, C. F. Enguix, 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.

[22] A. Garland and R. Alterman, "Learning procedural knowledge to better coordinate," in *IJCAI*, 2001, pp. 1073–1083. [Online]. Available: citeseer.ist.psu.edu/446179.html

[23] M. Beetz, *Concurrent Reactive Plans: Anticipating and Forestalling Execution Failures*, ser. Lecture Notes in Computer Science. Berlin: Springer, 2000, vol. 1772.

[24] S. Karim, L. Sonenberg, and A.-H. Tan, "A hybrid architecture combining reactive plan execution and reactive learning," in *9th Biennial Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, 2006.

[25] A.-H. Tan, "FALCON: A Fusion Architecture for Learning, COgnition, and Navigation," in *Proceedings of International Joint Conference on Neural Networks*, Budapest, 2004.

[26] B. Subagdja, I. Rahwan, and L. Sonenberg, "Learning as abductive deliberations," in *9th Biennial Pacific Rim International Conference on Artificial Intelligence (PRICAI)*, 2006.

[27] G. Gigerenzer and R. Selten, Eds., *Bounded Rationality: The Adaptive Toolbox*. Cambridge: MIT Press, 2001.

[28] L. R. Medsker, *Hybrid Intelligent Systems*. Kluwer Academic Publishers,, 1995, second Printing, 1998.