8-2006

A hybrid architecture combining reactive plan execution and reactive learning

Samin KARIM
University of Melbourne

Liz SONENBERG
University of Melbourne

Ah-Hwee TAN
Singapore Management University, ahtan@smu.edu.sg

A Hybrid Architecture Combining Reactive Plan Execution and Reactive Learning

Samin Karim[1], Liz Sonenberg[1], and Ah-Hwee Tan[2]

[1] Department of Information Systems
University of Melbourne
111 Barry St Carlton 3053, Melbourne, Australia
[2] School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798

Abstract. Developing software agents has been complicated by the problem of how knowledge should be represented and used. Many researchers have identified that agents need not require the use of complex representations, but in many cases suffice to use "the world" as their representation. However, the problem of introspection, both by the agents themselves and by (human) domain experts, requires a knowledge representation with a higher level of abstraction that is more 'understandable'. Learning and adaptation in agents has traditionally required knowledge to be represented at an arbitrary, low-level of abstraction. We seek to create an agent that has the capability of learning as well as utilising knowledge represented at a higher level of abstraction.

We firstly explore a reactive learner (FALCON) and reactive plan execution engine based on BDI (JACK) through experiments and analysis. We then describe an architecture we have developed that *combines* the BDI framework to the low-level reinforcement learner and present promising results from experiments using our minefield navigation domain.

1 Introduction

For many applications, agents require the ability to learn and adapt, as well as utilise an adequate knowledge representation that offers engineering benefits by virtue of a high-level, semantic representation. One of the major tradeoffs between an agent that represents its knowledge using a low level of abstraction, to an agent that represents its knowledge using a high level of abstraction, is in the type of representation used. 'Low-level' agents are more adept at learning and reactive behaviour [14], but usually lack a semantically rich knowledge representation, whereas 'high-level' agents have a semantically rich knowledge representation, but traditionally lack adequate learning capabilities that are grounded to this knowledge base. A hybrid architecture that combines these disparate knowledge representations may be the solution to achieve a unified agent that can learn as well as represent knowledge at a higher level of abstraction that carries greater semantic clarity.

Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 200–211, 2006.

In this paper we describe and evaluate the behaviours of a high-level reasoner, JACK (based on the BDI framework [3]), and a reinforcement learner (RL), FALCON (based on an extension of Adaptive Resonance Theory (ART) [5]), in order to demonstrate the aforementioned tradeoffs and to set the stage for the development of a new hybrid architecture that *combines* these approaches. The symbolic approach based on BDI has advantages in that it has a more precise, semantically rich knowledge representation that promotes effective introspection over the knowledge. Conversely, the RL approach (FALCON) can extract the equivalent control knowledge without manual intervention, but is disadvantaged by a less accessible representation (ie. low-level of abstraction).

The approaches that we employ have their roots in cognitive science. The Belief-Desire-Intention (BDI) approach derives from folk psychology, and provides a platform for human modelling and logical reasoning. BDI systems encode goal directed behaviours by using action sequences, or plans, derived from domain expert knowledge, usually specified at design time, about the task domain.

In modern cognitive science, many have held the view that cognition is a process deeply rooted in the body's interaction with the world [4]. The FALCON model [17], an instance of reinforcement learning [10] that stems from this philosophy, learns a policy by creating *cognitive codes*, which essentially are *rules* that associate current states to actions that lead to desirable outcomes (rewards). The strategy is similar to that adopted by the complementary reinforcement backpropagation algorithm [1].

Thus we see complementary approaches to solve a task. On one hand we have an *explicitly represented* procedural knowledge base (plans) utilised by the high-level BDI agent, and which are defined at design-time by the domain expert. On the other hand we observe an *implicitly represented* knowledge base accessed by the low-level FALCON agent and specified during repeated execution and experience gathering trials. By studying these different approaches to solving the same task, and analysing the tradeoffs of each, we should be well positioned to propose a *hybrid architecture* that combines these two dichotomous approaches.

To begin our investigation we have chosen a relatively simple task, which we label the 'minefield navigation domain'. It is similar to the one developed at the Naval Research Laboratory (NRL) [7], and involves an autonomous vehicle (AV) learning to navigate through obstacles to reach a stationary target (goal) within a specified number of steps. Our study indicates that, separately, FALCON and BDI agents can achieve very good performance in terms of success rates. However, the knowledge used by the two agents is vastly different. This points to many issues and challenges in developing an integrated architecture that performs both deliberative planning and reactive skill learning.

The rest of the paper is organised as follows. Section 2 provides details on the minefield navigation task domain. Section 3.1 gives a summary of the reactive FALCON model for this domain and presents the experimental results. Section 3.2 presents the BDI approach to the minefield navigation problem and associated experimental results. In section 4, a preliminary design and implementation that combines these two approaches is discussed and analysed. Outcomes and

discussion of experiments we ran using all three systems is covered in section 5. Section 6 describes related work, and finally section 7 summarises and discusses outcomes, highlights more general issues and points to future work.

2 The Minefield Navigation Task

We chose a domain that has sufficient regularity to make unsupervised learning and a priori defined plans viable. At the same time, there should be sufficient uncertainty and complexity to make re-evaluation and/or change of plans/actions necessary. The *minefield navigation task* (Figure 1) requires an autonomous vehicle (AV) starting at a randomly chosen position in the field to navigate through the minefield to a randomly selected target position in a specified time frame without hitting a mine. A trial ends when the system reaches the target (success), hits a mine or runs out of time (failure).

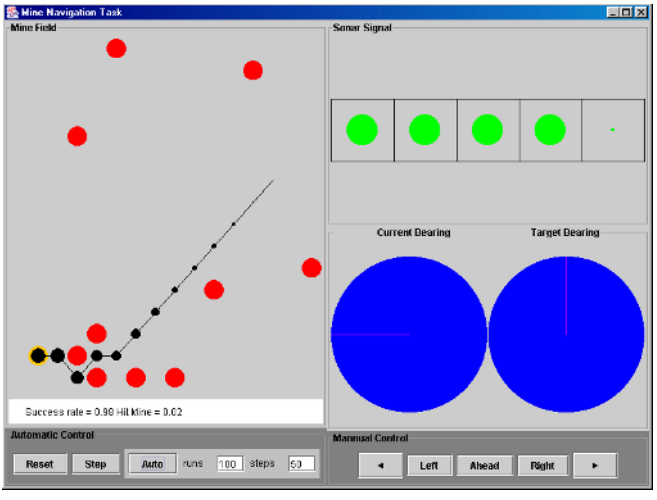

Fig. 1. The minefield navigation simulator

The system has a rather coarse sensory capability with a 180 degree forward view based on five sonar sensors. For each direction i, the sonar signal is measured by $s_i = \frac{1}{1+d_i}$, where d_i is the distance to an obstacle (that can be a mine or the boundary of the minefield) in the i direction. Other input attributes include the range and the bearing of the target from the current position. In each step, the system can choose one of five possible actions: `MoveLeft`, `MoveFrontLeft`, `MoveFront`, `MoveFrontRight`, and `MoveRight`.

The complexity for learning the minefield navigation problem is largely determined by the dimension of the sensory(state) representation. The state space is $S^5 \times B$ where $S = [0, 1]$ is the range of the sonar signals in the five directions and $B = \{0, 1, \ldots, 7\}$ is the set of possible target bearings.

3 Two Different Approaches

We initially employed two different approaches to solve the minefield navigation task, which are described in the following sections.

3.1 The Reactive Learning Approach: FALCON

FALCON is an extension of predictive Adaptive Resonance Theory (ART) networks
[6,16]. For reinforcement learning, FALCON makes use of a 3-channel architecture, consisting of a sensory field F_1^{c1} for representing the current state, an action field F_1^{c2} for representing the available actions, a reward field F_1^{c3} for representing the values of the feedback received from the environment, and a cognitive field F_2^c for encoding the relations among the values in the three input channels The reactive FALCON model acquires an action policy directly by learning the mapping from the current states to the desirable actions. The interested reader is directed to [17] for details on FALCON. It is sufficient to highlight here that FALCON achieves standard RL outcomes. As we will see later in section 5, FALCON is a good example of how low-level learners can solve complex tasks without the extraction and use of higher level abstract knowledge.

3.2 The Plan Execution Approach: BDI

The basic definition of BDI agency is described in [13]. We chose to use JACK as our agent-oriented programming platform, which supports BDI concepts and functionality. Architecturally, JACK consists of several constructs: agents, capabilities, events, plans and beliefs (see Figure 2). *Agents* are at the highest level of abstraction, and represent entities with *autonomous behaviour* within the system. Plans are executed when a relevant event occurs, which are posted as a result of many reasons, mainly from goals becoming active, a percept trigger, or from within other plans. Plans are specified by a domain expert at design time to negotiate general scenarios, as was the case in our implementation for the minefield domain. A detailed description of the JACK system can be found in [11].

4 A Combined Approach: The FALCON-BDI Hybrid

4.1 Towards a Unified, Learning Agent

An important goal of agent research is to develop an architecture that is able to learn and operate in real time. The RL mechanism of FALCON allows it to adapt its functional behaviour reasonably well but has inherent difficulties in representing and acquiring high level abstract knowledge such as those possible in BDI systems. Moreover, complex constructs such as context, action sequences and goals are not explicit features of low-level learners such as FALCON. Conversely, the BDI approach does not suffer from a nonintuitive representation such as the

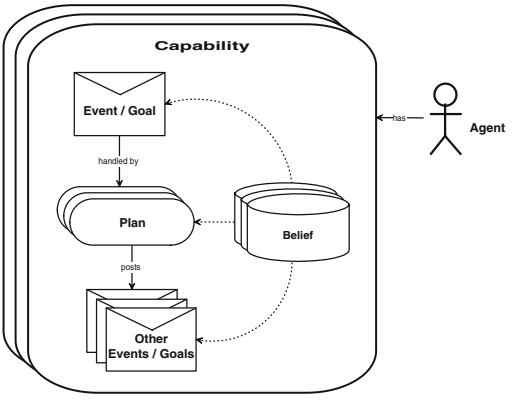

Fig. 2. JACK agent-programming language constructs

RL approach. However, the BDI approach requires significant domain expert input and is not inherently conducive to learning techniques due to it's high-level representation.

Due to these inherent limitations of each of these approaches, *combining* the plan-based, BDI approach with FALCON is an area we have tackled as part of an ongoing research effort. An approach is to have a *layered system*, with FALCON at the bottom level and the BDI-type system at a higher level, and their outputs appropriately combined. To this end, one possibility is to use the bottom level to *abstract concepts* that are used in the higher, BDI level. Abstraction of concepts from continuous feature values helps to produce higher level *symbolic knowledge*. Two different approaches where explored to achieve the abstraction of concepts. One approach was to abstract certain states to symbolic knowledge.

Another way to combine the reactive and plan-based approaches is to incorporate learning into BDI systems. JACK (BDI) plans are designed specifically for the task they were designed for, and can achieve almost perfect performance. However, in a less familiar environment, learning capabilities become desirable. There are several possible ways learning can take place in a BDI architecture.

1. Existing/predefined plans in a BDI architecture can be refined through the agent's subsequent experience with the environment. For example, the context of plans can be refined based on a rewarding strategy. The execution and actions of plans can be refined in a similar manner.
2. New plans can be learned in a BDI architecture. There are different approaches that could achieve this, such as: predefined knowledge that facilitates the learning of new plans (eg. pre-wired knowledge can be used to prime the learning process in, say, monitoring subsets of attributes or situations), or a low-level learner whose action recommendations are collated into plans.

The use of plans as opposed to rules (codes) results in a knowledge representation that has engineering advantages in that it is more comprehendible to humans.

Also, *policy compression* [14] is attained using plans (ie. the same information occupies less storage space). The approach described in the second point above is the basis for the architecture presented in the following sections.

4.2 Plan Generation from FALCON Codes

In this section we present our first step towards a unified architecture that merges the disparate approaches described in sections 3.1 and 3.2. The model consists of a BDI top-level (JACK) and RL bottom-level (FALCON). The resulting architecture *generates plans* via the BDI top-level from rules learned by the bottom-level. The crucial element of the system is that *a priori information* (specified by the domain expert) is used by the BDI top-level to assist in the generation of plans.

For our minefield navigation domain, plans consist of sequences of actions. The human domain expert supplies a priori data in the form of two agent goals that are associated to information pertaining to these goals. This a priori data can be likened to *clues* used by the BDI top-level for plan generation. The essential idea is that goals encapsulate a priori information that allows the BDI module to *converge* relevant FALCON action recommendations into plans:

1. **Goal:** Avoid mines.
   Clue: Consider when the agent is adjacent to a mine, meaning one or more sonar distance is equal to one.
2. **Goal:** Reach target.
   Clue: Consider when the agent is moving towards the target, meaning the distance to target is reducing from the previous step.

Plans will start being generated when these relevant conditions occur, as governed by the goals which encapsulate them. The ultimate effect is that actions are 'recorded' into plans. For the first goal, plans will start being generated when the agent is 'close to' a mine (ie. sonar reading indicates mine is adjacent to the agent). Similarly for the second goal, once the agent begins reducing its distance from the target, plans relevant to this goal will start being generated. So, after multiple runs, the BDI module will have accumulated a plan library of *condition* $\rightarrow$ *plan* profiles that can *subsume* the FALCON module when, for example, the utility (what ever this measurement may be) of the BDI plan is higher than the utility of the FALCON action. Plans which are executed repeatedly with success or failure will have their *confidence values* reinforced/penalised accordingly. For this purpose, the rewards in FALCON are utilised for adjusting the relevant plan's confidence values.

4.3 The Design

The architecture is illustrated in Figure 3. The top-left box (highlighted) is the critical module of the architecture, the *plan generation subcomponent* (PGS), which integrates FALCON to BDI. PGS generates plans by utilising a priori data in the form of "clues" that are part of specified goals. The actions, which are part

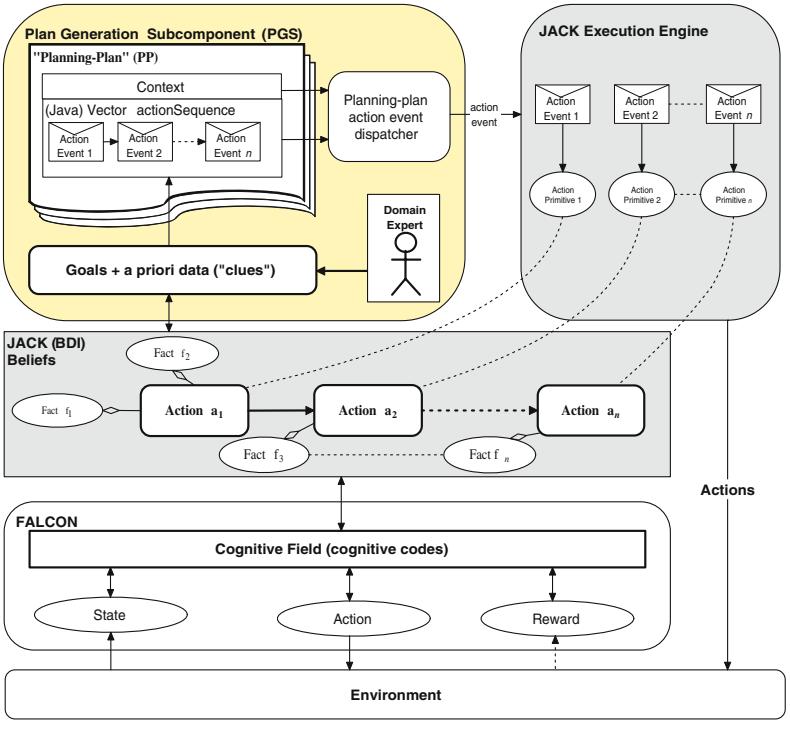

Fig. 3. FALCON–BDI architecture

of plans, are *grounded* to symbolic knowledge in the BDI belief base. Actions can either be performed by the BDI level or the FALCON level.

Algorithm 1 outlines the overall heuristic for plan generation and interaction between the BDI and FALCON modules. In this algorithm, plans are generated when an existing plan is not currently being executed and when FALCON actions are executed. When a plan is executing, FALCON actions are suppressed and plans are executed to completion. Actions and plans are selected on the basis of respective utility functions. Also, note that $Utility(a) = reward(a)$, and $Utility(p) = Utility(a_l) - Utility(a_1)$ (where a_l is the last step of a plan and a_1 is the first step of a plan - see step 6 of Algorithm 1).

5 Experiments and Results

We ran experiments using the domain described in section 2. All the systems described in the preceding sections were trained for 1000 trials, where each trial involved the random placement of 10 mines, the agent starting point and the target within a 16×16 grid. In each trial, the AV repeats the cycles of sense, act (and learn in the case of the FALCON and FALCON-BDI systems), until it reaches the target, hits a mine, or exceeds 30 sense-act(-learn) cycles. The same

Algorithm 1. The PGS plan generation and FALCON-BDI integration heuristic

Require: $n > threshold_1$, where $\mathcal{C}$ is the set of FALCON cognitive codes, and $\mathcal{C} = \{C_1, C_2, ..., C_n\}$

for each consecutive state-action tuple occurrence, (s, a_f), generated by FALCON **do**
  Update set of 'active goals', $\mathcal{G}$ (ie. goals relevant to current situation denoted by (s, a_f)).
  for each active goal $G_i \in \mathcal{G}$ **do**
    Calculate $Utility(a_f)$ and $Utlity(p_s)$
    if not already executing a plan **then**
      Select a plan $p_s : p_s = \{a_1, a_2, ..., a_c, ..., a_l\}$, where a_c is the current step of p_s, and a_l is the last step of p_s
    end if
    if $Utility(p_s) > Utility(a_f) \vee$ already executing a plan **then**
      Execute a_c, where $a_c \in p_s$
    else if $Utility(p_s) < Utility(a_f)$ **then**
      if there is an existing plan, p_e, in the $CustomPlan$ library, which (s, a_f) can be appended to **then**
        Append a_f to the end of p_e
      else
        Create a new $CustomPlan$, p_{new}
        Add a_f as first step of $p_{new} : p_{new} = \{a_f\}$
      end if
      Execute a_f
    end if
  end for
end for

Table 1. The success rates and hit mines ratio of the different systems with stationary and randomly moving mines. Number of trials = 1000 and maximum step allowance = 30 steps.

System	Mine Movement (success rate/hit mines ratio)	
	Stationary	Random
BDI	100% / 0.0%	85.7% / 10.7%
FALCON	89.4% / 8.5%	80.7% / 16.8%
FALCON-BDI hybrid	88.6% / 9.2%	78.7% / 18.8%

set of 1000 randomly generated minefield configuration was used across all experiments.

These results show that the BDI implementation, as expected, yielded superior performance due to the domain expert defined control knowledge base. The FALCON implementation followed BDI, and marginally outperformed the FALCON-BDI hybrid system. The results of FALCON and FALCON-BDI hybrid systems, however, are very similar, which shows much promise. We also ran the same experiments with random moving mines. In this instance, the agent must

Goal: "Avoid Mines"
Context: Mines to 'hard' left, left, and straight ahead; Target bearing = North-East, Agent's bearing = North-West
Plan body/Action sequence: Right → Hard right → Right → Straight → Straight → Straight

Fig. 4. Example PGS-generated plan

negotiate mines which move randomly within the grid. This presents the agent with a more complex task and hence proved to be considerably more difficult to solve, as evidenced by the performance decrease across all systems.

It is worth reiterating here that quantitative results alone are not the only component worth considering. The quality of plans and the aforementioned engineering advantages of the BDI plan representation (refer to section 4) are favourable outcomes not attained by low-level representations such as the rule-based representation of Falcon. An example plan generated by PGS is shown in Figure 4. Goals and the action sequence, or plan, are features of the BDI representation not present in the rule-based representation. To represent the same information in a rule-based representation would require separate rules for each step of the plan sequence. Unlike the BDI representation, the relationship between each step in a sequence of actions is not implicit in the rule-based representation.

6 Related Work

Sun [14] described a two-level model, known as Clarion, for learning reactive plans and extracting plans from reinforcement learners. The first three layers of the bottom level form a backpropagation network learning and computing Q-values and the fourth layer (the top level with only one node) determines probabilistically the action to be performed Clarion is an example of a hybrid architecture consisting of explicit (high-level) and implicit (low-level) representations.

Independently, Heinze et al [9] demonstrated the synergistic coupling of a machine learning architecture (CLARET) with a BDI top-layer in a hybrid architecture that achieves intention recognition of aircraft. CLARET was used to recognise spatial trajectories of aircraft and other observable objects. The BDI layer processed these observations with higher-level, goal-directed reasoning.

Sun and Sessions [15] mention that scalability is an issue for classical reinforcement learners. For a relatively complex domain, abstraction of attributes or function approximators are often needed. As Sun and Sessions's system made use of a radial basis function network or multilayer perceptron network, the system may not be able to learn and operate in *realtime*. However, a key strength of Sun and Sessions is the capability to perform *probabilistic planning*.

Also taking a bottom-up approach, the *DyKnow* framework by Doherty and Heintz [8] was essentially a signal-to-symbol transformer that, amongst other features, continually monitored perceived signals and created *higher-level cognitive*

objects facilitated by domain expert defined hypothesis tests. Temporal knowledge (plans) were among these higher level abstractions acquired, and are maintained as *chronologies*.

The Soar architecture [12] also achieves planning from a machine learning bottom-level by way of the well-known chunking mechanism. The PRODIGY architecture [18] combines a planning and reasoning framework with learning capability. The learning operates in three distinct ways: improving plan search efficiency, improving/adding planning operators (which are generalised atomic planning actions), and improving plan quality according to a predetermined metric.

These approaches and architectures have the common goal (amongst other goals) of *symbol grounding* high-level representations to low-level representations. Similarly, in this paper we presented an approach that *ties* a low-level RL module to a higher-level BDI module, essentially for the purpose of learning plans. However, the architecture presented in this paper is unique to current planning-learning hybrid systems in the literature with regard to the way plans are generated and with the assistance of domain expert knowledge, as well as the fact it is acquired in real-time.

7 Conclusion and Future Work

We have investigated ways in which different cognitively styled agents using knowledge representations of varying levels of abstraction can be combined into a hybrid architecture. Using a 'minefield navigation' domain, we firstly compared the performance and characteristics of a low-level reactive learner, FALCON, which is based on a reinforcement learning technique, to that of a high-level plan execution engine, JACK, that is based on the BDI framework. Secondly, we studied the way in which knowledge is represented and utilised in these two approaches, and then finally suggested a hybrid architecture that combines these two approaches to form a unified agent model.

The experiments reported have been designed to tease out issues relating to what knowledge to use and represent in our chosen domain. We have looked at four levels of knowledge representation: (i) the BDI approach (JACK) that supports the most expressive account of domain knowledge - procedures/plans are provided by a domain expert, in a rich language where that expertise "guarantees" task success; (ii) the reinforcement learner (FALCON) that works on primitive data where the system designer has provided minimal explicit guidance to the run-time engine, relying on the domain regularity to inform the learning, and (iii) a *hybrid architecture* that combines the plan-based and rule-based, or more generically, high-level and low-level representations, respectively, to achieve a system that is able to learn whilst maintaining the learned knowledge in a more abstract, higher-level BDI representation that is understandable to humans.

It is worthwhile noting that the hybrid architecture, presented in section 4, demonstrates a generalisable 'interaction' between different representations.

In this paper we present a BDI top-level that generates plans derived from an RL bottom-level. However, the bottom-level could quite easily be another type of module rather than the RL FALCON module (or more generally, a learning module) that is based on a representation with a similar low level of abstraction as FALCON. Furthermore, the application of this architecture goes beyond mobile robotics. For instance, plan generation, or sequence learning, has application in the biology domain [2].

It is hoped that further development of the hybrid architecture, whose preliminary design and evaluation is presented in this paper, will yield even better results in the future. Among consideration are improved plan selection, generation and management heuristics, as well as a more defined integration between the bottom and top levels. There are many avenues to consider in future. Diverse low-level learners other than FALCON will be considered, and as a major variation, the use of a human controller in place of the bottom-level will also be explored. Different knowledge rich domains where such a hybrid system would yield engineering and performance advantages will also be developed and investigated.

References

1. D. Ackley and M. Littman. Generalizaion and scaling in reinforcement learning. In *Advances in Neural Information Processing Systems 2*, pages 550–557, 1990.
2. P. Baldi, S. Brunak, P. Frasconi, G. Pollastri, and G. Soda. *Bidirectional Dynamics for Protein Secondary Structure Prediction*, volume 1828. Springer, Jan. 2001.
3. M. Bratman, D. Israel, and M. Pollack. Plans and resource-bounded practical reasoning. *Computational Intelligence*, 4(4):349–355, 1988.
4. R. Brooks. *Cambrian Intelligence: The Early History of the New AI*. MIT Press (A Bradford Book), 1999.
5. G. A. Carpenter and S. Grossberg. A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, June 1987.
6. G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D. B. Rosen. Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, 3:698–713, 1992.
7. D. Gordan and D. Subramanian. A cognitive model of learning to navigate. In *Nineteenth Annual Conference of the Cognitive Science Society*, 1997.
8. F. Heintz and P. Doherty. DyKnow: A Framework for Processing Dynamic Knowledge and Object Structures in Autonomous Systems. In *Intl. Workshop on Monitoring, Security and Rescue Techniques in MAS*, 2004.
9. C. Heinze, S. Goss, and A. Pearce. Plan Recognition in Military Simulation: Incorporating Machine Learning with Intelligent Agents. In *Proceedings, IJCAI Workshop on Team Behaviour and Plan Recognition*, pages 53–63, Stockholm, Sweden, 1999.
10. L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
11. S. Karim and C. Heinze. Experiences with the Design and Implementation of an Agent-based Autonomous UAV Controller. In *Proccedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, University of Utrecht, The Netherlands, 2005.

12. A. Newell. *Unified Theories of Cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.
13. A. S. Rao and M. P. Georgeff. BDI-agents: from theory to practice. In *Proceedings, First International Conference on Multiagent Systems*, San Francisco, 1995.
14. R. Sun. *Duality of the mind: A bottom-up approach toward cognition*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, 2002.
15. R. Sun and C. Sessions. Learning plans without a priori knowledge. *Adaptive Behavior*, 8(3/4):225–254, 2000.
16. A. H. Tan. Adaptive Resonance Associative Map. *Neural Networks*, 8(3):437–446, 1995.
17. A. H. Tan. FALCON: A fusion architecture for learning, cognition, and navigation. In *Proceedings, IJCNN, Budapest*, pages 3297–3302, 2004.
18. M. Veloso, J. Carbonell, A. Pérez, D. Borrajo, E. Fink, and J. Blythe. Integrating Planning and Learning: The PRODIGY Architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1):81–120, 1995.