A Connectionist Model of Planning via Back-chaining Search

Max Garagnani
Department of Computing
The Open University
Milton Keynes, MK7 6AA - UK
M.Garagnani@Open.ac.uk

Lokendra Shastri and **Carter Wendelken**
The International Computer Science Institute
Berkeley, CA 94704 USA
Shastri@ICSI.Berkeley.EDU
CarterW@ICSI.Berkeley.EDU

Abstract

A connectionist model for emergent planning behavior is proposed. The model demonstrates that a simple planning schema, acting in concert with two general purpose cognitive functionalities, namely, episodic memory and perception, can solve a restricted class of planning problems by backchaining from the goal to the current state. In spite of its simple structure, the schema can search for and execute plans involving multiple steps. We discuss how this simple model can be extended into a more powerful and expressive planning system by incorporating additional control and memory structures.

Introduction

Consider a classical planning problem, specified by an initial state, a goal state and a set of operators. A direct approach to solving this problem consists of searching the state space to find a 'path' between the initial and final states. Several symbolic planning systems adopting this approach in conjunction with the use of heuristics (Hoffman & Nebel, 2001; Haslum & Geffner, 2000; Bacchus & Teh, 1998) have recently shown notable improvements in efficiency on various benchmark problems.

Although a state space search algorithm is conceptually simple, it is not obvious how the data structures and control mechanisms required for the specification and execution of such an algorithm can be realized in a neurally plausible manner. In this paper we propose a connectionist model that exhibits a state-space search behavior. The model uses only a few simple control structures in conjunction with essential cognitive faculties, such as episodic memory, semantic memory, and perception.

Episodic memory (Tulving, 1995) refers to our ability to remember specific events and situations in our daily lives. The use of memory and experience in planning and reasoning has been investigated by several researchers (see (Waltz, 1995; Spalazzi, 2001) for useful accounts). Neurological and psychological data strongly suggests that episodic memory is distinct both in its functional characteristics and neural basis from other forms of memories such as semantic memory, memory for common sense knowledge, and procedural knowledge. It has been argued that events and situations in episodic memory are best viewed as *relational instances* that specify a set of bindings between the roles of a relational schema and objects that fill these roles in a given event or situation (Shastri, 2001b; 2002). We assume that a planning agent is capable of remembering past events such as "performing action A under conditions P lead to consequences C". Each episodic memory trace of this type can be represented as a triple of the form *Preconditions, Action, Consequences*, and we will refer to such triples as PAC memories (or events)[1].

Finally, we assume that the planning agent is capable of observing the current world state through perception. By this we mean that the agent can determine whether or not certain perceptually salient and directly observable relations hold in the world. For example, in the context of the classical blocks world scenario, this assumes that the agent can look at the table and determine whether or not a specific block is 'clear.'

A memory-based planning schema

Figure 1 shows the abstract structure of the proposed planning schema. In order to explain its behavior, let us describe the functionality of each of its components and their interactions using a simple example.

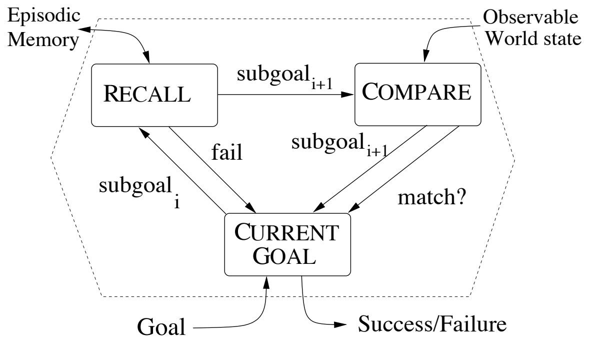

Figure 1: A block diagram showing the basic components of the planning mechanism.

Consider a planning problem with only two blocks ('A' and 'B') where the agent's goal G is to achieve `On(A,B)` (i.e. block A on block B) and the current world state is `On(B,A)`. Let us assume that the agent's episodic memory contains the two PAC events E_1 and E_2:

[1]Note that preconditions and consequences may contain multiple predicates.

E_1) In state P_1={`On(B,A)`},
action A_1= `Unstack(B,A)`
led to C_1={`OnTable(A)`,`OnTable(B)`};

E_2) In state P_2={`OnTable(A)`,`OnTable(B)`},
action A_2= `Stack(A,B)`
led to C_2={`On(A,B)`}.

(1) Initialize

The schema activity is initialized by conveying the goal G ($\equiv$ `On(A,B)` in our example) as input to CURRENT GOAL [2]. In the absence of any incoming activity from COMPARE, CURRENT GOAL simply passes control along with the goal (subgoal $_i = G$) to the RECALL component. The behavior of the CURRENT GOAL component in the presence of an input from COMPARE is different, and is described below.

(2) Recall

RECALL is activated when it receives the subgoal $_i$ (`On(A,B)`, in the example) as input from CURRENT GOAL. The function of this component is to search episodic memory for an event wherein a specific *action* (say, A) performed under some specific *preconditions* (say, P) lead to a set of *consequences* (say, C) in which subgoal $_i$ is true. In our example, E_2 happens to be such an event. When a matching event is found, action A and preconditions P are recollected and become 'active'; P becomes the current focus of the agent's attention[3], and control is transferred to COMPARE, along with P as the current subgoal $_{i+1}$. In the example, subgoal $_{i+1}$ = P_2 ={`OnTable(A)`,`OnTable(B)`}. If there are no events whose consequences 'match' the subgoal $_i$, the schema execution halts and signals a *failure*.

(3) Compare

The COMPARE block compares subgoal $_{i+1}$ with the current world state, which is assumed to be observable through perception. It returns a positive outcome *iff* subgoal $_{i+1}$ is true in the current state. In the example, the response is negative, as the world is still `On(B,A)`. After the comparison, the outcome and the subgoal $_{i+1}$ (=P) are passed to CURRENT GOAL, which takes control of the activity and reacts as explained below.

(4) Repeat

If CURRENT GOAL receives a negative result from COMPARE, the following happens:

- the original goal G is no longer passed as input to RECALL, and ceases driving the activity of the schema;
- the set of preconditions P (=subgoal $_{i+1}$) becomes the new subgoal $_i$ and is passed to RECALL by the CURRENT GOAL component.

[2]It is assumed that this goal is represented in other networks outside the planning schema and communicated to the schema via controlled spreading activation.

[3]In order to achieve the original goal G, it suffices to achieve P and execute action A.

At this point, one loop is completed and the procedure repeats from step (2) with P as the current goal.

If CURRENT GOAL receives a positive input from COMPARE, the schema terminates returning *Success* and the control is given to an appropriate 'Action schema' that will carry out the action currently active in memory (A). If the original goal G is not achieved via the execution of this action, the planning schema is re-invoked and re-initialized with G. Note that we are not assuming the existence of a working memory which would allow the agent to dynamically store sub-goals during the planning process or to maintain active more than *one* PAC event at a time. Because of this, the proposed system is forced to 're-discover' parts of the same plan every time an action is executed, as described below.

Returning to the example, the result of COMPARE is negative and no action is performed: G is 'forgotten', while P_2 becomes the new subgoal $_i$ and is passed on to RECALL. The schema now queries the episodic memory by asking if {`OnTable(A)`,`OnTable(B)`} has been achieved in the past: PAC event E_1 is recollected. The precondition P_1={`On(B,A)`} – required to perform `Unstack(B,A)` – becomes the new focus of attention and is compared with the current world state, producing a successful outcome: a chain of (two) PAC events connecting the goal to the initial state has been found, and the planning problem has been (potentially) solved. However, because of the absence of a working memory able to dynamically store goals and subgoals, all the agent can 'see' at this point is the last PAC event recollected. The currently active action (A_1=`Unstack(B,A)`), though, can and *should* be executed, since this will get the current state one step closer to the goal. After the positive outcome of COMPARE, the schema terminates returning '*Success*': the agent carries out the currently active action `Unstack(B,A)`, and the new state of the world becomes {`OnTable(A)`,`OnTable(B)`}.

After the action has been completed, the agent is 're-exposed' to the initial goal G (which has not been achieved yet), and the planning schema is re-invoked. The subsequent flow of activity is identical to the first part of the previous one, except that now P_2={`OnTable(A)`,`OnTable(B)`} matches the current state of the world, and thus action A_2= `Stack(A,B)` is executed. This leads to achieving the original goal G, and the schema is no longer invoked.

The connectionist planning schema

The planning mechanism described above has been implemented using the representational machinery of SHRUTI, a neurally plausible structured connectionist architecture that demonstrates how a network of neuron-like elements can encode a large body of structured knowledge and perform a variety of inferences within a few hundred milliseconds (Shastri & Ajjanagadde, 1993; Shastri, 1999; Shastri & Wendelken, 2000).

SHRUTI suggests that the encoding of relational information (frames, predicates, and schemas) is mediated by neural circuits composed of *focal-clusters*, and that the dynamic representation and communication of relational *instances* involves the transient propagation of *rhythmic* activity across these clusters. A role-entity binding is represented in this rhythmic activity by the *synchronous* firing of appropriate cells. Rules are encoded by links that enable the propagation of rhythmic activity across focal clusters, and a fact in long-term memory is a temporal pattern matching circuit.

In the past, SHRUTI's representational machinery has been used to encode commonsense knowledge (Shastri & Ajjanagadde, 1993), causal models (Shastri & Wendelken, 2000), as well as action schemas and reactive plans (Shastri, Grannes, Narayanan & Feldman, 1997) and decision-making (Wendelken, 2001).

A memory-based proto-planner

Consider the network structure depicted in Figure 2. This network fragment consists of two 'control' focal-clusters `ACHIEVE` and `RECALL`, two predicate focal-clusters `On` and `OnTable`, an action focal-cluster `Unstack`, two entity focal-clusters `A` and `B`, and a type focal-cluster `Block`. Typically, a focal-cluster contains several *control* and *role* nodes. For example, the focal-cluster `ACHIEVE` contains control nodes +, -, and ?, and role nodes I and G (the entity and type focal-clusters do not contain role nodes).

Role nodes within the focal-cluster of a predicate or schema provide a mechanism for expressing role (or parameter) bindings. In particular, a dynamic binding between a role and its filler is expressed by the synchronous firing of the role node and the focal-cluster of the object filling the role. A relational focal-cluster with bound role nodes designates a particular *relational instance*.

The *enabler* (?) node associated with a focal-cluster may be viewed as an "initiate query" or "initiate activity" node. In contrast, *collector* nodes (+ and –) associated with a focal-cluster indicate the outcome of a query or of other activity pertaining to the focal-cluster. In particular, the activation of the + (–) collector indicates a positive (negative) response to a query or signals a successful (unsuccessful) completion of some activity.

Inference occurs via the propagation of activity between focal-clusters. The links between the enabler and role nodes of interconnected focal-clusters allow queries posed in one focal-cluster to propagate to other focal-clusters: if role node R_1 is linked to role node R_2, the firing of R_1 induces synchronous firing in R_2, allowing dynamic binding propagation.

A query is communicated to a focal-cluster by activating its enabler node and binding its role nodes to appropriate role fillers. In Figure 2, the query "Can block B be placed on the table, given that B is on A?" is communicated by activating `?:ACHIEVE`, and synchronizing the firing of `ACHIEVE.I` and `+:On`; the firing of `ACHIEVE.G` and `+:OnTable`; the firing of `On.x`, `OnTable.x` and `?:B`; and that of `On.y` and `?:A`. The activity of `ACHIEVE` propagates to the `RECALL` cluster, resulting in the query "Is there some action which led to `OnTable(B)` from the precondition `On(B,A)`."

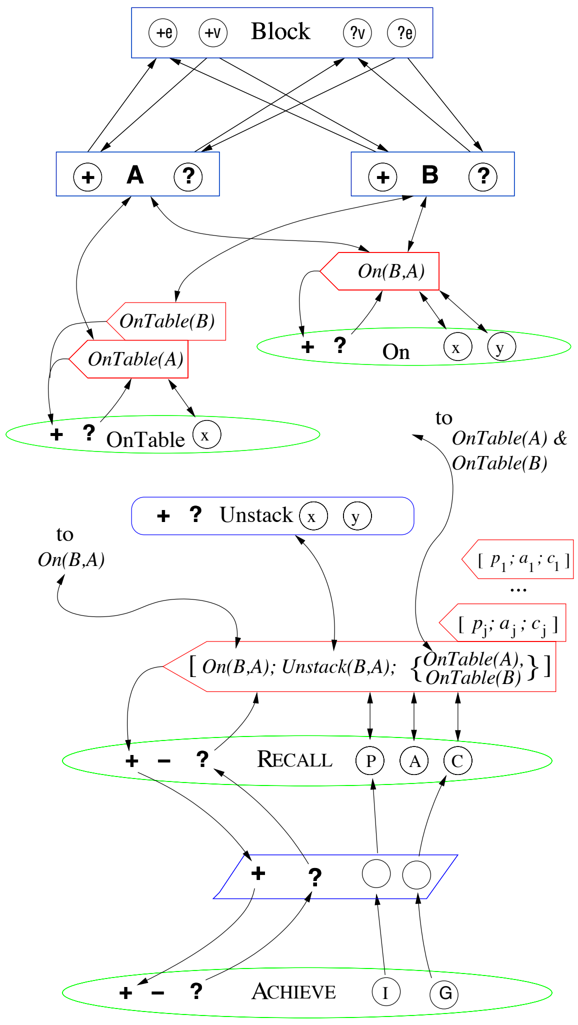

Figure 2: diagram showing network structure for a proto-planning mechanism.

Fact structures attached to a relational focal-cluster encode specific instances of that relation. If the query active at a focal-cluster matches an attached fact, the fact becomes active and, in turn, activates the positive collector of the relation's focal-cluster, binding (via synchronous firing) each of the relation's role nodes to the entity filling these roles in the fact. A neurally plausible model of how this might happen in the brain is described in (Shastri, 2001b).

In the case of `RECALL`, the facts structures attached to this focal-cluster represent episodic memories of specific PAC events. The activation of the query "Is there some action which led to `OnTable(B)` from the precondition `On(B,A)`" matches the memorized PAC event and leads to this event becoming active (i.e., recalled). This in turn results in the activation of `+:RECALL` and the synchronous firing of the unbound role `RECALL.A` with the representation of `Unstack(B,A)`: the system recalls that performing Unstack(B,A) when On(B,A) was true lead to OnTable(B) being true.

An important feature of the system consists of its ability to treat a relational instance as a role-filler (e.g. ACHIEVE.I $\equiv$ On(B,A)). In order to support this requirement, SHRUTI allows for two levels of temporal synchrony. Bindings between standard role nodes and entity/type nodes are represented within a rapid *minor* oscillatory cycle, while bindings between specialized role nodes and relational instances are encoded within a slower *major* oscillatory cycle.

The simple schema described above, consisting of the ACHIEVE and RECALL clusters acting in concert with the episodic memory, can retrieve previously memorized '*if-then*' (PAC) tuples. Thus, this schema can be construed as a *proto-planner* capable of returning one-step 'plans.' The next section demonstrates how this schema can search for *sequences* of actions, therefore constituting the next 'stage of evolution' of this proto-planner.

The planning schema in SHRUTI

Figure 3 shows how the planning schema of Figure 1 has been implemented using SHRUTI's representational machinery. It is easy to see how the focal clusters of this schema can be mapped to the elements of Figure 1 (the CURRENT GOAL block has been realized with two clusters, PLAN and SUBGOAL). We shall use the same example adopted earlier to illustrate how the schema can perform a basic form of *planning as search*.

Let us assume that the memory of the agent (represented only abstractly in the figure) contains the two PAC events of the previous example, namely, E_1 = (P_1, A_1, C_1) and E_2 = (P_2, A_2, C_2), and that the PERCEPTION block, when queried with input P, activates the + or – collector depending on whether the event bound to P is true or false in the observed world state.

The schema is invoked by activating the PLAN cluster's enabler ('?') node and by binding its role node G to the relational instance expressing the current goal (On(A,B) in the example). After initialization, activity propagates upwards along links to clusters SUBGOAL and RECALL. After few major cycles, ?:RECALL is activated, with role C firing in synchrony with the current goal {On(A,B)}. The PAC event E_2 matches the activity in the cluster and is retrieved. Consequently, +:RECALL becomes active, and the roles A and P are instantiated with actions A_2 = Stack(A,B) and relational instance P_2 = {OnTable(A,B)}, respectively (the clusters corresponding to predicate 'OnTable' and action 'Stack' are not shown in the figure). The activity of P reaches COMPARE. Since OnTable(A,B) is not true in the current world state, -:Compare becomes active. This leads to the inhibition of the links from PLAN to SUBGOAL, which blocks the propagation of the query PLAN(On(A,B)) through the schema. Simultaneously, activity from COMPARE reaches SUBGOAL: the role node G starts firing in synchrony with P, which was temporally bound to the relational instance {OnTable(A,B)}. Hence, this becomes the new (sub)goal of the schema, and its focus of attention. Activity from ?:SUBGOAL reaches ?:RECALL again, while role node C starts firing in synchrony with G. This leads to the retrieval of PAC event E_1, and hence, the precondition P_1 = {On(A,B)} gets bound to role P and role A is bound to the action instance A_1 = Unstack(B,A). These bindings are in turn propagated to COMPARE.

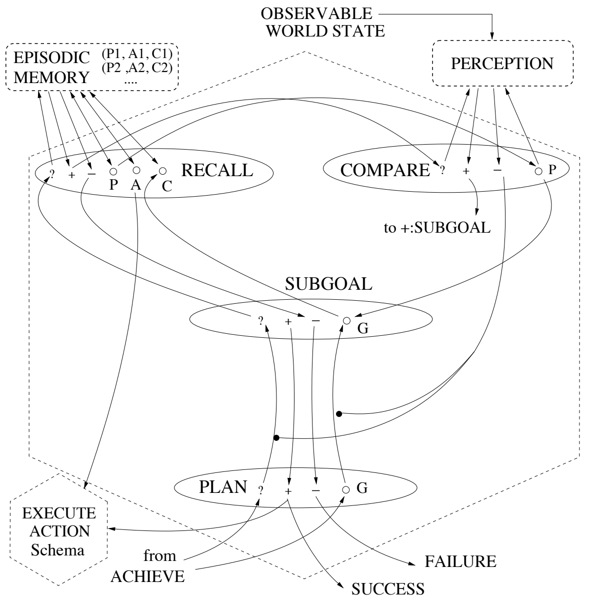

Figure 3: A diagram showing the connectionist structure of the planning schema.

The positive outcome of the comparison leads to the activation of +:COMPARE, which communicates to PLAN that the search has terminated successfully. After action A_1 is executed, the initial goal G={On(A,B)} (not yet achieved and still present in the system) causes the schema to restart. The subsequent flow of activation is identical to the initial part of the previous sequence, except that when P_2 = {OnTable(A,B)} is compared with the current state, the outcome is positive and the currently active action (Stack(A,B)) is executed. This achieves the goal G and terminates the activity.

Simulation results

The above planning schema has been realised and tested using the "SHRUTI Agent Simulator" software written in Java. The example described in the previous section has been used to test the functioning of the schema. Figure 4 shows the detailed trace of activation resulting from the actual simulation. Note how the diagram reflects closely the flow of activity described before, up to the first positive outcome of COMPARE.

Consider, for example, time point α of the diagram. Here, the PAC fact E_2 has just become active because of the initial query 'PLAN(On(A,B))', which has been propagated upwards and has led to the query 'RECALL(x,y,On(A,B))'. As a consequence, the two preconditions {`OnTable(A),OnTable(B)`} are about to become active, and will be propagated to the `COMPARE` cluster, where they will be matched against the current state of the world[4].

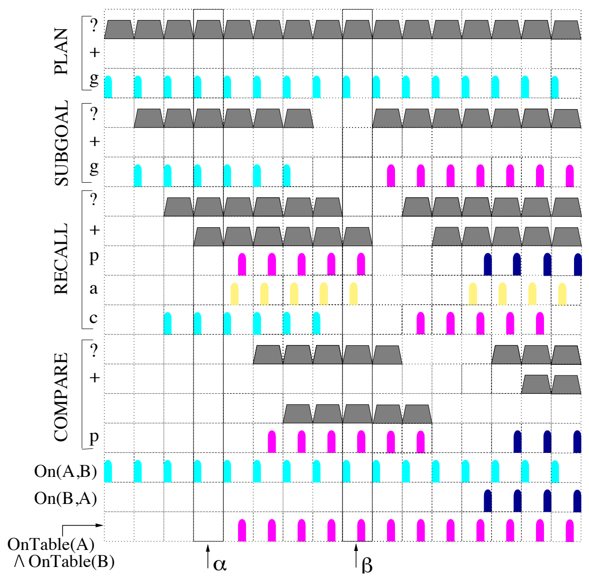

Figure 4: Node activation trace of the simulation.

Time point β is a snapshot of the situation immediately following the negative outcome of the (simulated) comparison. Notice how the flow of activity going from the `PLAN` cluster to the `RECALL` cluster has been interrupted by the negative outcome of `COMPARE` in order to allow the new goal {`OnTable(A),OnTable(B)`} to make its way through to cluster `RECALL`.

Discussion

The work described in this paper is part of a larger effort whose goal is to develop a neurally plausible architecture for reasoning, remembering, planning, and decision making. This paper presents progress along an important dimension of this ongoing effort. Perhaps the most interesting aspect of this work is the demonstration that general purpose cognitive faculties such as episodic memory, semantic memory and perception can be harnessed to produce a state-space search behavior and solve a subclass of planning problems.

The planning schema discussed in this paper is limited in a number of ways; however, as discussed below, this schema can be extended into a much more powerful and expressive planning system by incorporating additional control and memory structures, and by leveraging the full representational and expressive power of SHRUTI.

The proposed planning schema is susceptible to getting trapped in deadends. As the system searches for a path from the goal to the initial state, it can get caught in a state that subsumes a set of conditions D which do not match the consequent of any PAC event in memory. There is, however, a simple three part solution to this problem. First, the agent detects that it has reached a deadend state (this is signaled by the activation of `-:RECALL`. Second, the agent memorizes that this path leads to a deadend in the context of the current problem. It can do so by memorizing the following episodic memory trace: ìwhen trying to achieve the goal G, instantiating a subgoal D leads to a deadendî. [5] Third, the agent restarts the search and at each step in the search process retrieves both PAC and DEADEND events that match the current subgoal. Any retrieved PAC event that is counterindicated by retrieved DEADEND event is ignored. Since the memorization of deadends prunes the potential search space, with suf cient practice, the agent may memorize a large number of deadend events and carry out a highly ef cient search.

Another limitation of the proposed planning schema is that it needs to traverse the same paths through the state space several times during the course of nding a plan. However, if the agent could remember the path traversed from the current goal to the initial state, it would not have to rediscover the same plan subsequences many times over: plan execution would involve traversing the memorized sequence of PAC events only once (in the reverse order) and executing the actions associated with each PAC event in the sequence. Note that remembering such a path can be viewed as memorizing a sequence of PAC events. Learning of event sequences is a well-known property of episodic memory, but it remains to be seen how the process of such on-line memorization of event sequences can be fully integrated with the on-line retrieval of previous episodic memories. Working memory mechanisms can also play a complementary role in such on-line memorization. Our current research addresses the functioning of episodic memory (Shastri, 2001b; 2002) as well as that of working memory, and we hope that the development of powerful episodic and working memory models will directly bene t future work in the development of planning schemas.

Since the proposed planning schema operates within the SHRUTI architecture, the full range of knowledge representation and reasoning capabilities of SHRUTI can be leveraged during planning. This includes representing and reasoning with commonsense (semantic) knowledge, causal models, type hierarchies, context-sensitive prior probabilities of events and estimated utility/value of world-states. Thus, general purpose domain knowledge as well as planning speci c knowledge can be seamlessly combined to support planning involving not just memory retrieval, but also inference.

The functionality of the current planning schema is

[4]The perceptual task of verifying whether some conditions hold in the current world state was simulated by manually activating the $+$ or $-$ collector of the cluster ëCompareí as appropriate.

[5]The representational machinery required to encode such DEADEND ìeventsî is similar to that required to encode PAC events: like PAC events, the episodic memory trace of DEADEND events also involves role- llers that are partial state-descriptions, speci ed by sets of conditions.

also limited by its inability to make use of goal decomposition. Imagine that the agent is trying to nd a plan for the goal $G_1 \& G_2$ given the world state I and the two PAC events PAC(I', a_1, G_1) and PAC(I'', a_2, G_2) in memory. The planning schema described in this paper will be unable to solve the composite goal G_1 & G_2, even though it will be able to solve each of the subgoals G_1 and G_2 if presented individually[6]. In order to deal with goal decomposition, the schema must (i) recognize that it can solve one of the subproblems using one of the PAC facts, (ii) pick the subproblem to be solved, (iii) note down the subproblem that it is deferring for now, (iv) nd a solution to the selected subproblem, (v) shift attention back to the deferred subproblem, and (vi) solve the deferred subproblem. A connectionist implementation of this algorithm would require a more complex schema (control structure) than the one described in the previous sections, together with the ability to remember deferred goals. The memory of deferred goals can take the form of working memory (if deferred goals have to be remembered for a few seconds) or episodic memory (if the goals have to be remembered over longer time periods).

Another area of ongoing research of direct relevance to the work described here concerns the representation of complex action schemas and plans. In past work, we have shown that parameterized schemas capable of dealing with partially ordered actions, conditional actions, concurrent and iterative actions, as well as compositional and hierarchical actions can be encoded using SHRUTIís representational machinery (Shastri et al., 1997). This makes us con dent that the more complex control structures required for encoding more sophisticated planning schemas would not present an insurmountable problem.

A key issue that remains open is the learning of appropriate control structures. We are investigating this question within the frameworks of spike-timing dependent synaptic plasticity (Wendelken & Shastri,2000; Song, Miller & Abbott, 2000) and recruitment learning based on long-term potentiation (Malenka & Nicoll, 1999; Shastri, 2001a).

Acknowledgments

This work was partially funded by NSF grants 9720398 and 9970890.

References

Bacchus, F., & Teh, Y. W. (1998). Making forward chaining relevant. *Proceedings of the Fourth International Conference on AI Planning Systems (AIPS 1998)* (pp. 54ñ61).

Haslum, P., & Geffner, H. (2000). Admissible Heuristics for Optimal Planning. *Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000)* (pp. 140ñ 149). Breckenridge, Colorado: AAAI Press.

Hoffmann, J., & Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research, 14*, 253ñ 302.

Malenka, R. C., & Nicoll, R. A. (1999). Long-term Potentiation - A Decade of Progress? *Nature, 285*, 1870ñ 1874.

Shastri, L. (1999). Advances in SHRUTI - a neurally motivated model of relational knowledge representation and rapid inference using temporal synchrony. *Applied Intelligence, 11*.

Shastri, L. (2001a). A Biological Grounding of Recruitment Learning and Vicinal Algorithms. In J. Austin, S. Wermter & D. Wilshaw (Eds.), *Emergent neural computational architectures based on neuroscience*. Springer-Verlag.

Shastri, L. (2001b). A computational model of episodic memory formation in the Hippocampal system. *Neurocomputing, 38-40*, 889ñ897.

Shastri, L. (2002). Episodic memory and cortico-hippocampal interactions. *Trends in Cognitive Sciences, 6(4)*, 162ñ168.

Shastri, L., & Ajjanagadde, V. (1993). From simple associations to systematic reasoning. *Behavioral and Brain Sciences, 16(3)*, 417ñ494.

Shastri, L., Grannes, D., Narayanan, S. & Feldman, J. (1997). A Connectionist Encoding of Parameterized Schemas and Reactive Plans. In G. Kraetzschmar and G. Palm (Eds.), *Hybrid Information Processing in Adaptive Autonomous Vehicles*. Springer-Verlag.

Shastri, L., & Wendelken, C. (2000). Seeking coherent explanations - a fusion of structured connectionism, temporal synchrony, and evidential reasoning. *Proceedings of the Twenty-Second Conference of the Cognitive Science Society*. Philadelphia.

Song, S., Miller, K., & Abbott, L. (2000). Competitive Hebbian Learning Through Spike-Timing Dependent Synaptic Plasticity. *Nature Neuroscience, 3*, 919ñ926.

Spalazzi, L. (2001) A Survey on Case-Based Planning. *Artificial Intelligence Review, 16(1)*, 3ñ36.

Tulving, E. (1995) Organization of Memory: Quo Vadis? In M.S. Gazzaniga (Ed.), *The Cognitive Neuroscience*. MIT Press.

Waltz, D.L. (1995) Memory-based reasoning. In: M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks*. MIT Press.

Wendelken, C., & Shastri, L. (2000). Probabilistic inference and learning in a connectionist causal network. *Proceedings of the Second International Symposium on Neural Computation*.

Wendelken, C. & Shastri, L. (2002). SHRUTI-agent: A structured connectionist model of decision-making. *Proceedings of the 24th Conference of the Cognitive Science Society*. Washington, D.C. August, 2002.

[6] That is, assuming that I' and I'' hold in state I, and that I'' also holds in the state resulting from performing action a_1 in state I.