AN ARCHITECTURE FOR EXPERIENTIAL LEARNING

Michael C. Mozer, Klaus P. Gross

Carnegie-Mellon University, Department of Computer Science
Pittsburgh, PA 15213

INTRODUCTION

Our aim in this paper is to propose a cognitive architecture for learning in a complex, reactive environment. We hope that the architecture is sufficiently general that many forms of learning can be studied, yet not so general that it fails to constrain the nature of the learning task.

The architecture is that of an intelligent organism residing in the World Modelers environment [Carbonell (this volume)]. The World Modelers environment is a discrete-time simulation of a world that obeys a set of simplified physical laws. At each time step, an organism can interact with the world via a *Sensory-Effector Interface*. It may receive visual, auditory, tactile, olfactory, and gustatory information from the sensory commands and may act upon the environment through the effector commands (e.g., apply-force and turn-head). These commands are innate, non-adaptive, and serve as the basis for all interactions with the environment.

The World Modelers environment imposes several constraints on the organism architecture. First, the environment is information rich; hence, the organism must filter and condense the information available to its senses. Second, the environment is sufficiently complex that it cannot be completely predicted; hence, the organism requires mechanisms for handling unexpected situations. Third, the environment is reactive; hence, support should be available for active information gathering. Fourth, the environment provides an organism with information gradually over time; hence, the representation of temporal structure is necessary.

MAJOR COMPONENTS OF THE ARCHITECTURE

The architecture is partitioned into ten basic components, shown in Figure 1 along with arrows indicating the primary flow of information. These components function as modules in the sense that the design and implementation of one is relatively independent of the others, and each operates in a relatively autonomous manner.

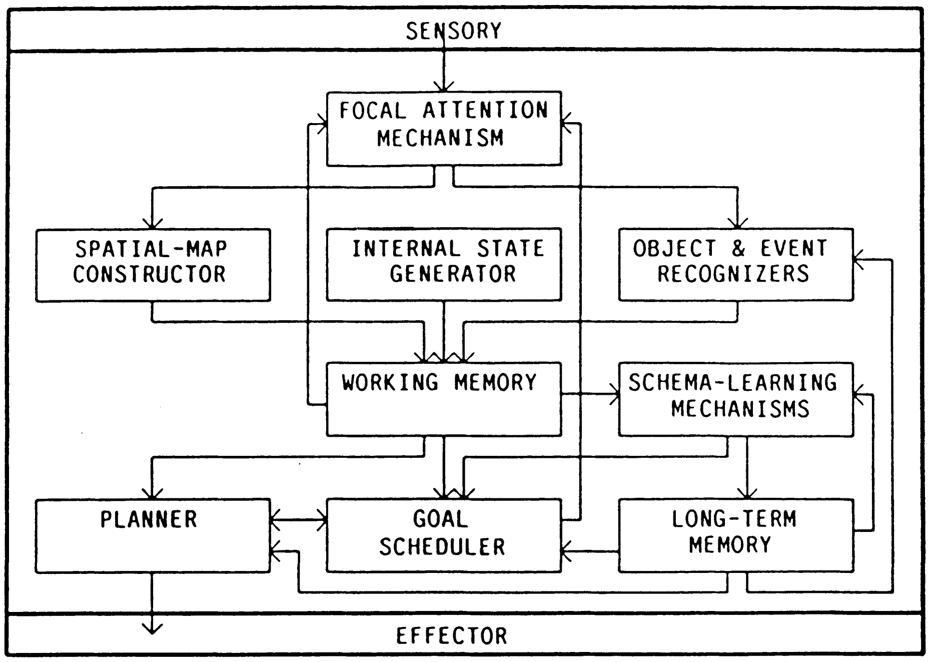

Figure 1: The Proposed Architecture

INTERNAL STATE GENERATOR

The Internal State Generator generates internal states of the organism (e.g., hunger, thirst, sickness, reproductive urges) based on the organism's physical characteristics (e.g., the quantity of food and drink it requires, the types of food and drink that are nourishing, the period of the reproductive cycle). These states simulate the physiology and innate drives of the organism. While external stimuli may have a bearing on these states, such stimuli do not trigger the states directly; for example, the quantity of food ingested affects whether the organism becomes satiated, but the satiated state is triggered by the organism's "stomach". All states that are derived from the external senses (e.g., pain as a result of tactile pressure and fear at seeing a tiger) are generated by the Object and Event Recognizers.

OBJECT AND EVENT RECOGNIZERS

The Object and Event Recognizers are responsible for producing descriptions of objects and events as perceived by the senses. These descriptions should be at a sufficiently high level that they may be used directly to suggest and formulate plans of action. In addition to directly-observable physical properties of an object or an event, these descriptions may include abstract properties such as the function of an object or the significance of an event.

All objects and events are interpreted in terms of existing knowledge structures. This knowledge is represented in the form of object and event *schemas*, which provide a framework into which object properties and event sequences are integrated. Schemas produce expectations (e.g., what color a banana should be or what an organism will do once it locates food). If these expectations are violated, a signal indicating the type and magnitude of violation will be sent to the Goal Scheduler.

We expect a high degree of interaction between Event and Object Recognizers. Each can help guide the search performed by the other. For example, a relevant event schema can focus the Object Recognizer on particular properties of an object. Similarly, properties of an identified object can focus the Event Recognizer on events involving objects with those properties.

SPATIAL-MAP CONSTRUCTOR

The Spatial-Map Constructor is responsible for creating a world-based description of the local environment (a *spatial map*). The spatial map contains information necessary to determine spatial relations among objects; for example, that food is a couple of feet behind the organism or that object x is located between objects y and z.

FOCAL ATTENTION MECHANISM

The Focal Attention Mechanism determines what is interesting in the world given: (1) expectations produced by schemas, (2) states produced by the Internal State Generator, (3) the organism's goals, and (4) learned or innate biases associated with particular objects, events, or regions of the spatial map. The Focal Attention Mechanism is necessary because of resource limitations on the organism which prevent the processing of all perceptual data. The focus of attention can be on a specific object, a particular spatial region in the visual field, or perhaps even on a particular set of schemas, which would allow the schemas to perform more effectively. The focus of attention must also specify a granularity at which objects and events should be perceived.

WORKING MEMORY

The Working Memory (WM) maintains a capacity-limited short-term store of (1) internal states of the organism, (2) recently recognized salient properties of objects, (3) recently recognized events, and (4) relations among objects. This information is provided by the Internal State Generator, Object Recognizer, Event Recognizer, and Spatial-Map Constructor, respectively. Because of capacity limitations on WM, the WM requires a memory manager to determine what information to displace as the memory fills. If a request is made to the WM for information that is not present in the memory, then the memory manager will set up a goal to discover the information. This information can be discovered either by refocusing attention, or by physically repositioning the organism so that the information will be directly perceivable.

LONG-TERM MEMORY

The Long-Term Memory (LTM) contains the organism's permanent, unlimited-capacity knowledge base. This knowledge includes object and event schemas, which can range from the specific, referring to a particular object or event, to the general, referring to a class of objects or events. Requests to LTM are in the form of partial descriptions of the desired information. For example, the Planner may request event schemas that satisfy a particular goal, or the Object Recognizer may request object schemas satisfying certain properties. Because the LTM knowledge base is so vast, a major design issue is how the knowledge should be organized.

GOAL SCHEDULER

The Goal Scheduler is responsible for determining the relative priorities of the organism's current goals, and for resolving conflicts among competing goals. In doing so, the Goal Scheduler provides centralized control over the various components of the architecture. The Goal Scheduler must also determine when a goal has been satisfied, at which point the goal is removed from consideration. The Goal Scheduler might include a learning mechanism that allowed for the acquisition of context-dependent goal priorities. For example, if it is easy to satisfy the *obtain-food* goal in warm weather and difficult in cold weather, the Goal Scheduler may learn to adjust the priority of the goal depending on weather conditions.

Goals of the organism come in one of several forms: (1) goals for satisfying an internal drive, (2) goals to handle expectation violations produced by the Object or Event Recognizer, (3) goals requesting information about the environment directed by the Working Memory, (4) goals created by the Planner in the course of attempting to achieve a higher-level goal, and (5) goals for exploratory behavior produced by particular learning mechanisms.

PLANNER

The Planner determines how to achieve the highest priority goals specified by the Goal Scheduler. Event schemas used by the Event

Recognizer can also be invoked for planning. An event schema, which specifies a sequence of sub-events, can be viewed as a plan for achieving the consequences of the event. The Planner must find an event schema or collection of event schemas whose consequences are the desired goals. The sub-events of a selected event schema in turn become subgoals of the plan, and the Planning process iterates. Ultimately, the Planner will send effector commands to the Sensory-Effector Interface.

SCHEMA-LEARNING MECHANISMS

While we do not want to characterize all learning as such, many forms of learning in our architecture will involve the acquisition and modification of schemas in LTM. Collectively, we call these the Schema-Learning Mechanisms. Consider one such mechanism, learning by imitation, the goal of which is to acquire a new event schema based on observations of another organism carrying out some action. (This event schema can later be used for performing the action.) The mechanism must determine *when* learning should take place and *which* subset of the objects and events in the environment are relevant to the learning task.

With regard to the "when" problem, our architecture offers several possibilities: learning might occur (1) when a violation of expectations occurs, signaling that properties of the environment cannot be accounted for by existing knowledge structures, hence new structures should be added; (2) when a state is observed in the environment that is related to a high-priority goal of the organism (e.g., another organism obtaining food), in which case the organism may wish to record how that state was reached; or (3) when a series of events involving attended objects is observed.

With regard to the "which" problem, the Focal Attention Mechanism and capacity limitations of WM offer a partial solution because they filter out much irrelevant information from the environment. The learning mechanism must still determine which states and events in WM are relevant, but this is a much reduced problem. It may appear that we have simply moved part of the filtering process to the Focal Attention Mechanism and Working Memory, but such a process is a general part of the architecture, necessary in many information processing tasks.

A UNIFORM PROCEDURAL REPRESENTATION

In the remainder of the paper, we focus on the role of event schemas in our architecture. Event schemas lie at the heart of the architecture because the procedural knowledge embodied in these schemas is useful to nearly all components of the architecture. Furthermore, event schemas provide a uniform representation of knowledge across components, which is essential for enabling efficient communication among the components and for facilitating interactions among various learning mechanisms.

EVENT REPRESENTATION

We represent an event by a triple composed of an *action*, a set of states that are true before the event takes place (the *before-states*) and a set of states that are true after the event takes place (the *after-states*). The action is a description of the event, consisting of an event name, the times delineating the event, object(s) involved in the event, and any other parameters not captured by the state changes. The before- and after-states capture only states that change during the course of the event and whose change can be attributed to the event.

EVENT-SCHEMA REPRESENTATION

Our representation of an event schema is described in [263]. To summarize briefly, the representation can be divided into six sections. The *schema-vars* section specifies internal state variables of the schema. The *update* section contains code for updating the schema-vars. The *init-states* section specifies a set of states that must be true for the schema to be applicable, in essence, a context in which the event occurs. The *goal-states* section specifies a set of states that must be true for the successful completion of the event. The *script* section specifies a sequence of of lower-level, more basic events (hereafter, *sub-events*) that compose the event. Finally, the *restrictions* section specifies either numeric or propositional restrictions on parameters of the sub-events, often in terms of the schema-vars. The script provides only a partial ordering on sub-events; precise temporal relations are specified via the restrictions section.

The script and restrictions sections suggest a conceptual partition of knowledge into structural and featural components. To see this, consider the *Rectilinear-motion* schema, which encodes knowledge of motion in a straight line. The script indicates that rectilinear motion of an object is composed of a sequence of location changes (each a primitive event), terminated when the object stops or collides with another object. The restrictions on the sub-events indicate that with each location change, the new location of the object must lie along the straight-line path formed by the previous locations.

EVENT SCHEMAS AND LEARNING

The partition of an event schema into script and restrictions sections suggests two distinct Schema-Learning Mechanisms. The mechanism involved with script learning must pay attention to the order in which events occur and the boundaries of an event; the mechanism involved with the learning of parameter restrictions, however, overlaps highly with the learning of object schemas. In fact, the parameters of an event provide a basis for grouping objects according to their functional role in an event. For example, suppose that org1 repeatedly observes org2 moving towards objects and then consuming them. After an initial observation, org1 may construct an *eat* schema, consisting primarily of a script of basic actions. To comprehend subsequent instances of org2's behavior in terms of the *eat* schema, org1 must note that every object eaten fills the same parameter slot of an event. A generalization mechanism that attempts to find the restriction on this parameter would also discover the concept of food.

PREDICTIVE NATURE OF EVENT SCHEMAS

In our implementation, event schemas are predictive. That is, an event schema can generate an exact prediction as to what sub-event should occur next, or the relative likelihood of a small number of sub-events. The prediction is specific to the point of indicating not only the general type of sub-event, but also the expected time of occurrence, objects involved, and resulting state changes. This expectation-based encoding allows event schemas to be used by many components of the architecture. We have found application for event schemas in (1) perceiving higher-order events, where the schema predictions are matched to observed events; (2) signaling novelty or unexpected events, as when a predicted event fails to occur, or when an observed event cannot be assimilated into any active schema; (3) planning actions, where the schema predictions are considered as actions to be executed; (4) monitoring the execution of a plan, which can be achieved by invoking a schema simultaneously in perception and in planning, and using the perceptual schema to check that the planner schema produces the desired effect; and (5) mental simulation, where the schema predictions are used to determine the effectiveness of a proposed plan.

FUTURE DIRECTIONS

We have attempted to convey a general flavor of our architecture, along with several examples of how the architecture, though fairly general, does help to define the nature of the learning task. The most valuable function of the architecture, however, is that it specifies a set of components that can be assumed by researchers interested in learning issues. These components have proven sufficiently useful that we are currently implementing and elaborating the architecture. Once individual learning mechanisms are understood in terms of this framework, we can attempt to integrate the mechanisms into a complete, intelligent organism.

ACKNOWLEDGMENTS

The ideas presented in this paper developed from discussions by the World Modelers Group at C-MU, in particular Jaime Carbonell, Keith Barnett, Greg Hood, Hans Tallis, and Alain Rappaport. This work was supported in part by ONR grants N00014-79-C-0661 and N0014-82-C-50767, DARPA contract number F33615-84-K-1520, and a grant from the System Development Foundation.