Specialized Learning and the Design of Intelligent Agents

Joanna Bryson

September 18, 1997

Abstract

The reactive approach to artificial intelligence has developed impressive demonstrations of life-like robots and simulations by breaking behavior into carefully engineered behavioral modules which react quickly to the environment due to specialized perception (Horswill 1997). The next step in scaling the complexity of reactive robots is to acknowledge and understand the role of state and learning in such systems. Learning is an integral part of perception and behavior; this article proposes it should also be specialized and modular. We present a system of thinking about state and learning, heuristics for determining the extent of specialization vs. adaptivity that should be used, and an example of an architecture that supports the design and development of such systems. We also briefly describe experiments in both simulated blocks world and mobile robotics.

1 Introduction

In some ways embodied agents appear significantly more intelligent now than they did a few decades ago. The reactive approach to robot intelligence (Steels 1994) allows us to create rapid, robust behaviors by replacing extensive planning and modeling with carefully engineered behaviors and continuous sensing of the environment (Matarić 1997, Maes 1990). However, in some ways our new agents look stupider than the robots of the 1970's such as Shakey (Nilsson 1984) and FREDDY (Ambler, Barrow, Brown, Burstall & Popplestone 1975). We no longer have demonstrations involving assembly and manipulation as well as map building and navigation. Essentially, we have robots with a very limited set of skills that are capable identifying when those skills should be applied. They cannot generate their own goals or learn new tasks; in short, they cannot perform complex behavior.

A common response to this situation has been to create hybrid robotic architectures that supplement reactive lower layers with conventional AI planners (Bonasso, Firby, Gat, Kortenkamp, Miller & Slack 1997, Hexmoor 1995, Levison 1996). Planning, however, has been at best difficult to scale and at worst provably intractable (Chapman 1987, Horswill 1997, Wooldridge & Jennings 1995). An alternative research path is to explore ways to expand the current envelope of competence in reactive robotics. We suggest that the problem of scaling the complexity of reactive robotics is not due to having reached a theoretical ceiling beyond which full-scale planning and symbolic thought are necessary. Rather, the problem is a consequence of current technique and methodology.

In particular, there is a need to focus on the use of learning in reactive architectures. The reactive robotics approach depends crucially on the practice of carefully engineering each behavior (Brooks 1991*a*, Wooldridge & Jennings 1995). Consequently, it has tended to deprecate the use of learning and the presence of state. Practitioners argue that relying on previous experience must necessarily reduce a robot's reactivity to the environment, exposing it to danger and missed opportunities in a dynamic world.

We would argue instead that learning is crucial to intelligent agents. Some amount of variable state is necessary to organize complex behavior regardless of representational issues. Variable state (hereafter referred to just as "state") and learning are inextricably linked: learning can be defined as the process of changing variable state.

In this article we present an analysis of state, and consider the problem of determining what kind of state is appropriate to which situations. We also consider the question of when learning is necessary, and what form it should take. The following two sections present a way of thinking about learning and state. In Section 2 we consider the lessons that can be taken from the biological literature. In Section 3 we consider the uses of state and learning in the field of reactive and behavior-based robotics. We then introduce an analysis of state, and present heuristics for using each type of state representation. These sections also introduce our own architecture, based on reactive hierarchical control and modular learning. The architecture's design reflects our analysis of state and learning, and experiments we have conducted illustrate it.

2 Learning in Animals

Animals are our primary working example of what we consider intelligence to be (Brooks 1991*a*, McGonigle 1991). Autonomous agent research more than most branches of artificial intelligence has always acknowledged the extent to which it exploits the solutions of natural intelligence (see for example Maes, Matarić, Meyer, Pollack & Wilson (1996)). This is particularly true of ethology, the study of animal behavior as an evolutionary derivative.

Earlier this century, behaviorists (animal researchers who concentrated on laboratory experiments), proposed that animals learn only through a general process of being able to create associations.

> The [general process assumption] position is that all learning is based on the capacity to form associations; there are general laws of learning that apply equally to all domains of stimuli, responses, and reinforcers; the more frequent the pairings between the elements to be associated, the stronger the associative strength; the more proximate the members of an association pair, the more likely the learning(Gallistel, Brown, Carey, Gelman & Keil 1991).

Learning by association, also called "conditioning", does appear to be a general learning mechanism with parameters that hold across species, presumably indicating a common underlying mechanism. However, behaviorist research itself eventually demonstrated that animals *cannot* learn to associate any stimulus with any response. Pigeons can learn to peck for food, but cannot learn to peck to avoid a shock. They can, however, learn to flap their wings to avoid a shock, but not for food (Hineline & Rachlin 1969). In related experiments, rats presented with "bad" water learned different cues for its badness depending on the consequences of drinking it. If drinking lead to shocks, they learned visual or auditory cues and if drinking lead to poisoning they learned taste or smell cues (Garcia & Koelling 1966).

These examples demonstrate highly specific, constrained and ecologically relevant learning mechanisms. For example, the content of the associations allowed for the rats biases their learning towards information more likely to be relevant: poison is often indicated by smell or taste, while acute pain is often the consequence of something that can be seen or heard. Such results were originally interpreted as constraints placed on open learning that might be dangerous, but research has since indicated the opposite. Specialized systems exist to form important associations. For example, poison avoidance in the rats is handled by a specific one-shot-learning mechanism in the olfactory section of their amygdala.

Gallistel et al. (1991) hypothesize that learning by an individual organism serves as a last resort for evolution. It is introduced only when behavior cannot be fully predetermined, because the competence involved requires flexibility on a less than evolutionary time scale[1]. Further, such learning is not necessarily associative. Barn owl chicks learn to calibrate acoustically the precise location of prey (necessary because it is dependent on the shape of the individual bird's head) and bees learn the ephemeris of the sun for navigation (dependent on season and latitude).

These examples suggest that animals may be born with limited units of variable state which are instantiated during development by observing the world. Variable instantiation can also take the form of perceptual learning or categorization. Vervet monkeys have three distinctive warning cries for predators which require different defensive action. These cries are dedicated to pythons, martial eagles, and leopards. Baby vervets make cries from a very early age, but across more general

[1]Evolutionary modeling research on the Baldwin Effect suggests that there is little evolutionary pressure to genetically hard code rules that are consistently learned (Hinton & Nowlan 1987).

objects. For example, they may give the "eagle" cry for anything in the sky, the "leopard" cry for any animal, the "python" cry for a stick on the ground. They are born attending to the sorts of stimuli they need to be aware of, but learn fine discrimination experientially (Seyfarth, Cheney & Marler 1980).

Although animal learning is specialized, individual elements are not necessarily constrained to a single purpose or behavior. Gallistel et al. (1991) that a single adaptive solution or mechanism may be leveraged by multiple processes once established. For example, foraging animals allocate foraging time between sites proportional to the relative amount of fodder available at those sites. Gallistel et al. (1991) suggest that the same mechanism an animal uses to localizing itself within its environment may be used for effectively modeling the forage availability.

3 Reactive and Behavior-Based Modeling of Intelligence

The most convincingly animal-like agents have typically been produced under the reactive and behavior based approaches to artificial intelligence (Blumberg 1996). However, so far none of them have exploited multiple, possibly interacting forms of specialized learning as the previous section suggests animals do. This section reviews the reactive and behavior-based approaches, the following introduces extensions that support the design of systems with such learning.

The reactive approach requires that an agent respond more or less directly to each situation, without the intervention of detailed deliberation or planning between sensation and action. Although this approach runs counter to many intuitive notions of rationality and intelligence, it has proved effective for problems ranging from navigation-based robot tasks (Connell 1990) to playing video games (Agre & Chapman 1987), modeling human perception and problem solving (Ballard, Hayhoe, Pook & Rao forthcoming).

Behavior-based AI refers to an approach inspired by Minsky (1986), where many small, relatively simple elements of intelligence act in parallel, each handling their own area of expertise (Brooks 1991*b*, Matarić 1997). In theory, these simpler elements are both easier to design and more plausible from an evolutionary standpoint. The apparent complexity of intelligent behavior arises from two sources: the interaction between multiple units running in parallel, and the complexity of the environment the units are reacting to.

Behavior-based systems are not necessarily reactive (Minsky (1986) intended planning and complex representation to be parts of the elements). Reactive systems, on the other hand, are likely to be behavior-based. Decomposing intelligence into such units decreases the number of actions any particular unit needs to select, and the number of steps necessary for that unit to move from sensation to action. Both of these allow the system to react faster.

The central design problem for behavior-based systems is behavior arbitration: determining which parallel module controls physical behavior at any one time. This is a problem not only for design but also for program development. It can be very difficult to determine if faulty behavior is the result of one or more behaviors operating simultaneously.

The central problem of reactive planning is the external Markov assumption — the notion that the next action can be entirely determined by external state. Most intelligent agents will often find themselves passing through identical environments as the result of different original initiatives that should require the agent to select different behaviors. For example, the offices in a laboratory where a robot operates may be connected by a single hallway. The hallway is essentially the same environment whichever office the robot needs to enter next.

4 Learning and State

In their struggle to minimize state, the architects of dominant reactive systems (Brooks 1991*b*, Maes 1989) have complicated both learning and control unnecessarily by confounding the flow of information with the flow of control. The methodology presented here divides the issues of control, or *when* a behavior is expressed, from perception, or *how* it is expressed. This section briefly introduces a typology of state, followed by heuristics for how a designer should select between the types for a given problem. The next section illustrates each type in considerably more detail, often with implementations.

4.1 Types of State

Learning and knowledge are expressed in different types of state which we can divide into four categories:

Control State State in the form of the currently active behavior or behaviors, and the paths that activation can follow. Control state records the immediate past in terms of action selection and determines, in combination with sensory information, the agent's next act. Conventional AI systems hold control state in a plan, in which a program pointer determines what the next step should be. Finite state machines, often used in behavior-based AI (Brooks 1991*b*), hold similar state.

Deictic Variables Fixed variables which refer to a particular object of attention. Deictic variables allow a system to generalize over cases where particular plans embodied in control state may operate. An example from ethology is the classic work on imprinting by Lorenz (Lorenz 1973): a gosling will follow and learn from whatever mother-shaped thing it sees (possibly a baby-buggy) during a critical period after hatching. Upon reaching adulthood, a male goose will try to mate with similar objects.

Perceptual Memory Specialized systems and representations where information can accumulate. Perceptual memory may last from a few cycles to the lifetime of the agent. Perception requires memory because very little information is available in each snapshot of sensory information.

Meta-State State about other internal state. Research in mainstream AI suggests that learning about one's own behavior patterns and developing meta-rules is necessary for complex, human-like behavior (Winston 1992). Certainly animal brains contain much recurrent wiring. This may be necessary for learning of complex patterns such as hierarchical behavior (McGonigle & Chalmers forthcoming).

4.2 Heuristics for Using Different Types of State

The list above is ordered in descending reactivity. The earlier items have the least amount of dependence on previous events. The basic assumption of reactive AI is that the less state a system has, the better. We share this assumption, emphasizing that learning should be as specialized and controlled as possible. It should be pointed out, however, that control programs are themselves state. Minimizing state may therefore mean *increasing* the amount of learning, since compact learning can replace control.

The design methodology presented here is based on the evolution of code through the corkscrew cycles of rapid prototyping and frequent testing (Bryson in press). Behaviors are to be built up individually where possible, while co-evolving the control and learning. This section provides heuristics for determining when to use each of the elements listed above.

- Control state is the basic element of this typology. It is not necessarily reactive, but reactive systems all have some element of control state, made reactive by using constant sensory monitoring to determine the branches between state. Control state does not represent the world, except indirectly. It must deal with its environment. Consequently, information about the environment can be deduced from the control structure.

- Deictic variables are set to represent or at least reference some particular object in or aspect of the environment. As such, they restrict attention and make control more manageable. They should be used when control structures would otherwise be replicated. Replication of control state is cumbersome and unreliable in terms of an evolving system, since multiple instantiations may need to have the same updating. It may also be expensive in terms of space.

- Excessive control state may also indicate the need for perceptual memory. Determining the appropriate context for a behavior must not require too many control steps, or the system will become complicated and slow. This situation indicates that the sensory information for determining the context should be consolidated into a representation. Such representations can be updated by parallel processes to the control structure, while the interface to the control system can be reduced to a single "sensory" check.
- Meta-state is necessary for an agent to learn from or redescribe its own experience or alter its own processes. Karmiloff-Smith (1992) proposes that rediscription of knowledge is central to human development. For example, the ability to learn a skill as a unit by rote then decompose its components for reuse in a different strategy. Norman & Shallice (1986) argue that deliberation is a way to bring increased attention to bear on routine processes, thus increasing their precision or reliability. Meta-state can be understood as essentially another form of perception, though it is not yet clear what conditions indicate a need for more of this kind state.

5 Examples of State Designs

Learning can be understood as any meaningful, persistent change in computational state. The previous section showed that state can appear in a variety of forms, facilitating highly specialized learning, in a variety of degrees of persistence and kinds of representation. This section illustrates the uses of each of the four kinds of state, using examples from the literature and experiments from our own work. The structure of our architecture, organized around modularized learning and reactive hierarchical control, is described as necessary. For more details of the architecture, see (Bryson in press).

5.1 Control State

One of the myths of artificial intelligence is that the early experimenters in reactive planning advocated stateless systems. In fact, the fundamental units of subsumption architecture (Brooks 1991*b*) are augmented finite state machines. Thus at each moment every behavior records a particular state of execution, and this state helps determine what that behavior will do next.

The most reactive behaviors do respond directly and continuously to factors in the environment. For example, in the first few Braitenburg Vehicles (Braitenberg 1984) motor speed was directly correlates to light or heat intensity. The Polly vision-based obstacle-avoidance algorithm also determined speed and direction by the apparent distance to obstacles in each individual visual frame. (Horswill 1993)).

However, even simple behaviors may require some kind of stored internal state, and often sequential control. For example, one of the most influential subsumption robots, Genghis (Angle 1989), would back off and turn away from obstacles detected by bumping its antennae. Notice that the environmental state to which the robot reacts is only detected while the antenna is bent, but the behavior determined by that state must be perpetuated over the period of backing up after the antenna is disengaged, and this behavior must be followed by a turning behavior that avoids returning to the same location.

In a conventional AI system, the initial collision would have resulted in an internal representation of the collision point as an assumed obstacle, and the registration of this knowledge would have resulted in a planner selecting a new behavior (such as exploring the obstacle) or a new trajectory. In Genghis, the collision is registered only temporarily in control state. One behavior is forced into a different state, which results in the different actions. Once normal behavior is resumed, the event will be forgotten.

This disposal of information may appear wasteful, but in practice the information is difficult to record accurately, and is often transient. The superiority of the reactive approach to local navigation has been shown empirically (Maes 1990, Bonasso et al. 1997). A more significant criticism, however, is that this approach exchanges the complexity of adequately representing the environment for complexity in designing the control.

5.1.1 Programming Control State

Programming under the original reactive architectures is notoriously difficult (Wooldridge & Jennings 1995). Programming difficulties have been part of the motivation for the two and three layer architectures (Bonasso et al. 1997) and for the situated planning approach in general (Levison 1996). There are two main reasons for this problem.

First, subsumption architecture and similar approaches have sensor information flow and control structure represented in the same connections between behaviors. Ideally in subsumption architecture, each behavior should receive sensor information directly, so that each behavior is itself reactive. In fact, many of the behaviors take their inputs from reading the outputs of other behaviors, and similarly many express their own behavior by inhibiting or suppressing (overwriting) the inputs or outputs of other behaviors. (See (Brooks 1989) for example.)

Trying to effect control and data flow through the same mechanisms leads to complications. One of the principles of subsumption architecture is that once a layer of competence is developed, it should never be altered for higher layers. In Herbert (Connell 1990), this principle broke down when a behavior was added to reach and grasp a soda can. The robot's arm broke its line of sight to the can, so the lower behavior that moved unless it saw a soda can would be activated in mid reach. Brooks and Connell solved this problem by adding timers to the behaviors (thus the *augmented* finite state machine) to allow for their short-term global suppression if one behavior needed to "take over".

This example also illustrates the second reason why programming is difficult in such systems — many behaviors, such as the steps in reaching for a soda can, are best described as elements of a sequence. Many systems based on ethological theories of motivation have systems of competing behaviors, where each behavior determines its own activation level (Blumberg 1996, Norman & Shallice 1986, Correia & Steiger-Garção 1995). However, such systems can also be difficult to program and to scale, even in situations where some explicit sequencing is set in terms of activation from precondition to dependency (Maes 1989).

5.1.2 Reactive Hierarchical Control

To address these issues we developed a control architecture. The architecture provides two primary types of control structures for organizing behaviors: action sequences referred to as LAPs (Learnable Action Patterns) and small reactive plans referred to as competences. An action sequence is built of actions and independent sense checks. Provided the actions do not fail catastrophically and the sensory information matches its checked value, a sequence will execute until completion. This sort of behavioral unit reduces the combinatorial complexity of action selection and provides some of the persistence needed to overcome the problems caused by the external Markov assumption.

Competences are prioritized collections of LAPs and other competences. The highest priority element is a LAP that recognizes the attainment of some goal. The other elements offer steps that tend to converge to that goal. Elements of the competence fire by activation level, initially equal to their priority. If an element is unable to execute completely, it briefly auto-inhibits to allow other lower priority elements an opportunity to fire. On completion, an element may also be habituated for a longer term. This 'boredom' mechanism lets the system use multiple strategies (via other elements) and allows the competence to fail, allowing control to shift to another competence. A competence fails if none of its elements can fire; an element with zero or less priority will never fire.

Sequences and competences express full and partial orderings of behaviors respectively. They also guarantee that only one behavior within the set is expressed at a time. A special form of competence called a drive allows its elements to execute in course-grained parallel. Drive elements habituate and recover at a fixed rate. Consequently a best-effort form of scheduling parallel behaviors is possible. The presence of drives and competences ensure the system is still reactive.

This system can be used to solve Herbert's dilemma. The grasp of the can will either be an action sequence or a competence[2] that is an element of the competence that searches for and retrieves cans. By making these competences part of a single drive element we ensure that they will never operate at the same time. Each drive element can be thought of as an attention resource. By having a

[2]An action sequence would be appropriate for the exact behavior in (Connell 1990).

control structure (whether a sequence or a competence) for the grasp, we remove the ambiguity of what failing to see the can means. If the robot is attending to the grasp process, then it has found the can it is grasping, and is not searching. If the can is lost (say, knocked away by a clumsy grasp) then when the grasping terminates unsuccessfully the search will resume.

The reactive hierarchical control architecture is capable of expressing loops and infinite chains of behavior, since competence elements can themselves be competences. No stack is maintained when a competence passes on control to an element that is a sub-competence. Rather, when that element (or one of its children) successfully completes, control returns to the initial competence in chain. If the world has not changed substantially, control will again be reactively passed through to the parent competence. If the world *has* changed, then these competences may not be called as the system reacts to a new environment.

This system may not be able to encode every possible subsumption architecture program. However, it should be capable of expressing any published *algorithm*, or intended behavior, that has been implemented in subsumption architecture. The architecture is designed to help the programmer realize the intended interrelations between behaviors in an intuitive way. Programmers tend to think about behavior in terms of causality and preconditions, priorities, sequences, and in goal-oriented clusters. These concepts are easily expressed in this architecture. At the same time, we make allowances for parallel activity at the level of the behavior (see the following section on perception).

The architecture cannot be used for some of the applications of the more ethologically oriented architectures. For example, Cooper, Shallice & Farringdon (1995) shows how the Norman & Shallice (1986) model breaks down similarly to humans given certain parameter values that are intended to model different pathological neurological conditions. These parameters are not accessible in our architecture. This reflects a tradeoff in favor of programmability and realistic execution speeds. The types of ethological research that can be conducted are found at a higher level of abstraction.

We have not implemented any experiments using strictly the control architecture. Our first results are described under the next section, involving the use of deictic variables.

5.2 Deictic Representation

One of the problems of traditional representation has been the number of items a system needs to represent, if it represents a complete world model. A solution to this problem that is gaining acceptance is the use of deictic variables (Ballard et al. forthcoming, Horswill 1997). A deictic variable is a permanent memory structure with changeable external reference that can be incorporated into plans. For example, representing a blocks world problem may require individually labeling each block. Deictic variables provide a restricted set of labels such as the-block-I'm-holding or even the-green-block-I-most-recently-saw.

Deictic variables were popularized in the work of Agre and Chapman (Agre & Chapman 1988, Chapman 1990) who used them to write a reactive system that successfully plays computer games. The ideas behind deictic variables can also be found in Minsky's pronemes (Minsky 1986), in visual attention Ullman (1984), and apparently in philosophical work of Heidegger (Dreyfus 1992). Whitehead (1992) uses deictic variables to simplify reinforcement learning. Horswill (1995) uses them in combination with a live visual image to produce a system that answers prolog-like queries ("Is there a green block on a red block?") about a scene using no other representation other than the visual signal itself.

The benefits deictic representations bring to reactive architectures are similar to what it brings to more traditional representations, though possibly move substantial. One of the common criticisms of reactive planning is that it transfers the complexity of data representation into control complexity. Thus the reduction of complexity brought about by using deictic variables for a reactive system is not in terms of amount of data, but in the size of the control architecture, the number of behaviors, or the number of reactive plans (Horswill 1997, Rhodes 1996).

5.2.1 Trials with the Reactive Hierarchical Control Architecture

Pure reactive planning is closely related to deictic representations. In the example given earlier of Genghis avoiding an obstacle, the obstacle might be perceived externally as the deictic variable the-thing-I-just-hit. In this case the representation is not encoded explicitly, it exists only in the state of the behaviors responsible for the avoidance. However, making deictic representations explicit requires little modification to the system described in Section 5.1.2.

Deictic representation requires two things. First, some state to be allocated for the variable. This does not necessarily have to be added at the architecture level. The reference of the variable may be what is in the hand, where the camera is centered, or what the vision system's tracking is fixated on. Thus in a real robot, the deictic variable state may be situated either in the external world (the place right in front of the robot), within the robot's mechanism, or it may refer to state in a peripheral system such as a visual routine processor. The second thing deictic representation requires are primitives for setting and sampling that variable.

The first implementation of the Hierarchical Control Architecture was a blocks world simulation called Braniff. The simulation was based on Whitehead (1992), a reinforcement learning thesis using visual routine theory. The primitive behaviors were assumed to be grounded completely[3] in commands of the Visual Routine Processor described in (Horswill 1995). Thus the behaviors themselves had no internal state beyond control. The state of the deictic variables was recorded in the simulated environment; in the robotic implementation that state would have been maintained by the VRP.

Braniff was located in a blocks world to address a challenge to behavior-based AI by Chris Malcolm. Malcolm's own research uses reactive, behavior-based primitives to support a planning system for assembly (Malcolm & Smithers 1990, Wilson 1996); he considers assembly too complex of a task for a strictly behavior-based system(Malcolm 1992). Whitehead's thesis was chosen in particular because of its status as a well-known advance of reinforcement learning. Reinforcement learning develops a look-up table for behavior selection based on observed state. Our intention was to compare our own system with behavior arbitration as results from reinforcement learning.

The system described in Whitehead's thesis learned to pick up a green block when presented with up two four stacks of up to four red, green and blue blocks. The system used only two visual markers (deictic variables); keeping this number low was necessary for the reinforcement learning, since reinforcement methods are highly susceptible to combinatoric explosion. The target green block is selected by exploiting a known human visual primitive for "popping up" elements of a visual scene with a simple descriptive criteria, such as color or shape. No attempt was made to optimize which green block was targeted for grasping. The system learned to first select a green block, then to move visual attention to the top of its stack to grasp and remove the top block, until the target block (or another, higher green block) was unburied and held in the hand.

Scripting this solution as a competence under Braniff was straightforward. Every element of the competence was habituated with no recovery, to guarantee failure in the case where no green blocks were available. Essentially, one marker is used to keep track of the selected green block, providing persistence. The other is used to find and manipulate the top of the stack, and to observe what is located in the hand. Observing the hand is necessary to detect goal completion. One critical programming strategy emerged from this experience: elements of the same competence must not swap state on trigger information. For example, one element should not move attention to the hand as its final task, while another moves it to the block. Triggers need to have canonical states, and elements that move triggers *out* of these states must move it back before completing. Of course, elements to move triggers *into* a canonical state are often a necessary part of goal completion.

The next task attempted was also taken from Whitehead (1992). Whitehead states that his initial goal for the simulation is to stack a red block on a green block. However, no results were presented beyond grasping red and green blocks. One reason for this became clear when coding the task: another significant location is necessary: the location of the stack. The state of the stack indicates the next task for the robot, whether to pick up a green block, a red block, or whether the stack is complete. In addition, without the mechanism of a subgoal, a fourth marker would be necessary. In order to gauge the state of the stack, a marker must check its contents. If that marker detects that the stack contains a green block, this can trigger the competence to grasp a green block, which uses both spare markers. If there were no separate competence, another marker would be needed to make constant comparisons to the top of the stack. This demonstrates an interchange of control state and deictic state, with considerable combinatorial ramifications.

[3]With the exception of the robot arm commands, which were quite unrealistically considered primitives both in Whitehead (1992) and our simulation.

As a final experiment, we constructed a copy demo similar to (Chapman 1989). The demo requires building a stack to match a goal stack present in the environment. Coding this more general task took no more steps than coding the red on green task. The extra complexity of tracing two stacks was compensated for by not having to represent the goal entirely in the control structure. The process of coding took less than three hours to write and debug.

5.3 Perception

Perception is a psychological abstraction for the level at which processed sensory information is used for cognition. The abstraction is clearest in humans: what we perceive is what we report, whereas what we sense is presumably more directly a consequence of the environment. Perception is dependent both on learning and on context (Carlson 1994). We can learn to discriminate differences without having any explicit knowledge of what those differences are. In some cases, this learning requires no feedback, simply exposure to the appropriate stimulus (Sundareswaran & Vaina 1994).

If, as suggested by behavior-based AI, intelligence should be programmed in specialized modules, then this tight coupling between learning and sensing implies memory should be as well. Though notice by "modular" we do not necessarily mean independent, but rather localized by behavior and representation. This is consistent with animal intelligence, where learning has been shown to be highly specialized and constrained.

The main contribution of reactive, behavior-based AI has been a change in the source of action selection for an intelligent agent. Plans relying on complex internal models of the world have been replaced by distributed actions relying directly on fast, cleverly specialized perceptual routines. However, the complexity of task that can be managed has been seriously reduced. We have argued in previous sections that better structured control state allows for greater behavior complexity, while still allowing for reactivity. Perceptual state definitionally reduces reactivity, since it relies on "old" information. However, it plays the same role as structured control state in that it preserves information already experienced to disambiguate control decisions.

We may create the same effect as perceptual state with an enormous action selection look-up table that includes events experienced at varying times before the choice point of each step in control as a part of the indexing. From this perspective, the use of perceptual learning can be seen as similar to the use of deictic variables. In fact, the more like a deictic variable perceptual learning can become, the better. Our thesis that intelligent agents can best be developed through the use of specialized learning modules is really a restatement of the well established result that learning is dependent on bias (Kaelbling 1997). We attempt to maximize bias by minimizing learning and constraining each necessary adaptive element individually.

5.3.1 Modular Learning and Reactive Hierarchical Control

Learning is applied when programming control is otherwise prohibitively complex. For example, if the triggers that discriminate the set of elements of a competence become too convoluted, either because of multiple dependencies or multiple steps to find those dependencies, it will be easier to reduce the triggers to one or two statements based on new perceptual state.

Besides the general assertion that learning should be used as little as possible, an obvious indication that it has been overused is if the agent cannot learn required perceptual information reliably. In this case, a further constraint is needed. Notice that the additional information for that constraint may also take the form of more perceptual state rather than more control.

The introduction of new perceptual state implies that some new behavior will be needed to compute this state apparently continuously, so that a value is available on demand. Our approach emphasizes the separate flow of control from the flow of data, but the continuous flow of data is not eliminated. (See Figure 5.3.1.) The reactive hierarchical control architecture is developed in conjunction with the behaviors it triggers. These behaviors may process sensory information directly, or in conjunction with learned perceptual state.

Perceptual learning was not required for Braniff and the assembly work described above, but was very necessary for implementing the next version of our system, Edmund, on a mobile robot. The first use of behaviors centered around state was simply to manage the information load from 32 range sensors (pairs of sonar and infra-red) and 3 drive velocities (translation, angle of drive,

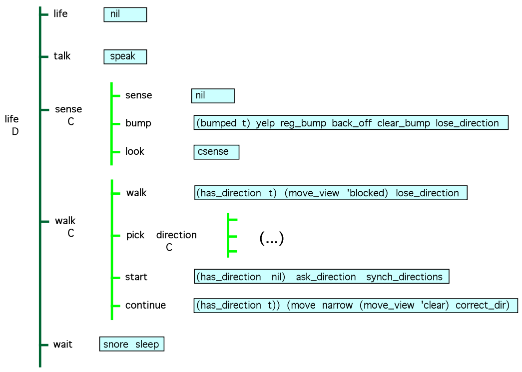

Figure 1: An example of a script written under the hierarchical reactive control architecture. This particular script allows a robot to build a map of appropriate behavior by interacting with the user, while simultaneously taking care to navigate and report critical information.)

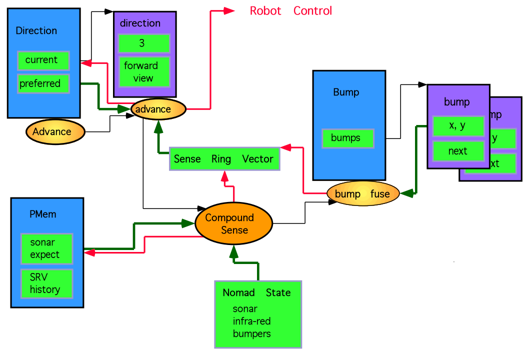

Figure 2: The modular learning and behavior library which supports the control script on the left. Behavior libraries are platform dependent, but can support many control scripts. Control determines when a behavior is expressed, perception controls how.

and turret rotation.) The two types of range sensors were consolidated into single values, indicating the nearest hazard in 16 directions. The intended directions of motion were limited to these 16. Each direction was given local state including an appropriate window into the sensor array. Also, deictic variables were set up to attend to two special directions, the currently active direction and the intended direction of motion.

Running the robot provided the first demand for more persistent perceptual memory. The sonar readings are frequently very noisy, so they were buffered with a behavior that guesses the next value from the previous few. This information has a lifespan of approximately half a second. Another problem is obstacles that cannot be detected by either range device (see Figure 3.) Bump data is stored for individual incidents, but both buffered sonar and bumper information are fused into the original sensor array, so that none of the motion behaviors that depended on it needed to be altered. Bump data is aged not by time, but by distance. It is both most relevant and most accurate while the robot is still within ten feet of where the impact occurred. Since the stored information is purely odometric, long term representation of bump events would need to have another representation.

Our next tasks were to provide the robot with more cognitive navigation abilities. First, the robot should be able to repeat a path it had learned by direction. This involves storing information on the contexts in which decisions (rather than default behavior) were made and recording the direction chosen. This memory is indexed only by rough location, and is open to revision and consolidation. In addition, we want the robot to be able to successfully select a direction from a number of possible strategies. This requires an episodic memory of recent events. The events themselves are the same type of elements as the mapping information, but their access structure is time rather than location dependent.

These examples illustrate several reasons why one element of memory may be distinct from another.

- The original signal is very different.
- The memory needs to record events that occur at different rates.
- The memory needs to decay at different rates.
- Patterns from the memory emerge at different rates.
- Different representations are most simple or minimal.

These issues constitute further evidence for the claim that learning should be specialized and modular.

Figure 3: The Nomad 200 robot running scripts written in Edmund. In this picture it is encountering a desk — the surface is too low for the sonar sensors to detect when proximate, and the leg too dark for the infra-red sensors, so the robot must learn about the desk with its bumpers.

5.4 Meta-Learning

The final form of state is state about other state. In the context of the reactive hierarchical control architecture and modularized behavior, meta-state can be reduced to two forms: state about control state, and state about perceptual memory.

Control state in Edmund divides into two pieces: activation levels and control structure. A great deal of work has been done on varying activation levels between behaviors. In particular, the ethological literature describes modules competing for attention by evaluating their own activation level; several ethologically inspired architectures do precisely this (Maes 1989, Correia & Steiger-Garção 1995, Blumberg 1996, Cooper et al. 1995). Blumberg (1996) in particular has achieved probably the most coherent behavior in this genre by combining a strategy of resource-based intelligent constraints and mutual inhibition between competing behaviors. Blumberg also provides a mechanism for reconnecting the flow of control in order to implement conditioning. Under our typology, this strategy would also be meta-learning, though conditioning could also be implemented on the perceptual level.

The action sequences and competences in the control architecture were designed to support another kind of meta-learning, possibly referred to as thought. The idea is to allow evolutionary-type processes to derive new behaviors or solutions when old ones cannot be produced. (See for a similar theory (Calvin 1996).) The simpler problems here are how to generate new thoughts and how to test and store a winning solution. The hard problems are how to tell when a new plan is needed, and how to recognize which solution wins.

Another application of meta-level control would be to implement Norman & Shallice (1986)'s notion of deliberate action. Norman and Shallice's theory holds that deliberate control is triggered by special circumstances. Deliberation then monitors and modifies routine processes, possibly by slowing them or providing additional sensory information.

Our current research involves perceptual state rather than control. We hypothesize that the learning of sequential tasks in animals involves the monitoring of memory, or a memory of memory. This assumes that episodic memory is not stored with chronological ordering. Recent events are clearer simply because of decay caused by interference, which may be aggravated by change in context. Neurophysiological research demonstrates that a change of context alters the indexing of the place cells in the hippocampus in rats (Wilson & McNaughton 1994). Recent psychological research indicates that human sequence learning can best be modeled by associating elements of the sequence graded relation to its start and end (Henson 1996). Consolidated motor sequence memory has particular cells dedicated to units of behavior and to motor transitions (Tanji 1996). Presumably, then, some element of memory allows for this transduction by observing episodic memory. We are currently experimenting with navigational systems built on this premise. The meta-state is simply further perceptual state.

Karmiloff-Smith (1992) has proposed that development of skills depends on learning new representations for existing behaviors. Hexmoor (1995) describes learning as redescription between skills stored in separate declarative and implicit levels of knowledge of behavior. In Edmund, these systems could learn from each other simply by storing perceptual state on each other's activities as they are relevant to the individual controlling behaviors.

Implementing these final forms of meta-memory requires no structural change to the Edmund architecture. Additional behaviors and memories are simply added into the behavioral library. Meta-state involving control would require some architectural changes as the representations for control are currently isolated from the behavioral library. However, the overall theoretical structure would remain substantially intact.

6 Summary: Designing Learning

In summary, developing complex intelligent behavior requires not only modularized behavior, but also modularized learning. Agent programs should constrain and specialize each learning task as much as possible. Control state records both static knowledge of procedure and dynamic decisions on current approach. Deictic variables simplify control state by providing a point of reference through which procedures can be generalized across contexts. Perceptual memory allows control to be determine by factors that appear only across time rather than in the immediate environment. Meta-state allows the agent to learn from and change its own behavior.

This article began with the assumption that an approach based on carefully engineered behavior, reactive artificial intelligence, is a more reasonable way to design the control of intelligent agents than planning-based approaches. We argued that such an approach is extremely dependent on a methodology for both programming and problem decomposition to enable the designers to create such agents. We presented a way to think about the use and implementation of learning as a key part to advancing the current state of the art for reactive systems. We presented a decomposition of intelligent state into the categories of control state, deictic variables, perception and meta-state. Finally, we illustrated this decomposition of memory, using an approach combining reactive hierarchical control and modular learning.

References

Agre, P. E. & Chapman, D. (1987), Pengi: An Implementation of a Theory of Activity, *in* 'Proceedings of the Sixth National Conference on Artificial Intelligence', Morgan Kaufmann Publishers, Inc., Seattle, Washington, pp. 196–201.

Agre, P. E. & Chapman, D. (1988), What are Plans For?, AI memo 1050, MIT, Cambridge, MA.

Ambler, A., Barrow, H., Brown, C., Burstall, R. & Popplestone, R. J. (1975), 'A versatile system for computer controlled assembly', *Artificial Intelligence* **6**(2), 215–218.

Angle, C. M. (1989), Genghis, a Six Legged Walking Robot, Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts. Bachelor's thesis.

Ballard, D. H., Hayhoe, M. M., Pook, P. K. & Rao, R. P. N. (forthcoming), 'Deictic codes for the embodiment of cognition', *Brain and Behavioral Sciences*.

Blumberg, B. M. (1996), Old Tricks, New Dogs: Ethology and Interactive Creatures, PhD thesis, Learning and Common Sense Section.

Bonasso, R. P., Firby, R. J., Gat, E., Kortenkamp, D., Miller, D. P. & Slack, M. G. (1997), 'Experiences with an architecture for intelligent, reactive agents', *Journal of Experimental & Theoretical Artificial Intelligence* **9**(2/3), 237–256.

Braitenberg, V. (1984), *Vehicles: Experiments in Synthetic Psychology*, MIT Press, Cambridge, Massachusetts.

Brooks, R. A. (1989), 'A Robot That Walks: Emergent Behavior from a Carefully Evolved Network', *Neural Computation* **1**(2), 253–262.

Brooks, R. A. (1991*a*), Intelligence Without Reason, *in* 'Proceedings of the 1991 International Joint Conference on Artificial Intelligence', pp. 569–595.

Brooks, R. A. (1991*b*), 'Intelligence without Representation', *Artificial Intelligence* **47**, 139–159.

Bryson, J. (in press), Agent Architecture as Object Oriented Design, *in* M. P. Singh, ed., 'The Fourth International Workshop on Agent Theories, Architectures, and Languages (ATAL97)', Springer-Verlag.

Calvin, W. H. (1996), *The Cerebral Code*, MIT Press.

Carlson, N. R. (1994), *Physiology of Behavior*, 5 edn, Allyn and Bacon, Boston.

Chapman, D. (1987), 'Planning for conjunctive goals', *Artificial Intelligence* **32**, 333–378.

Chapman, D. (1989), 'Penguins Can Make Cake', *AI Magazine* **10**(4), 51–60.

Chapman, D. (1990), Vision, Instruction, and Action, Technical Report 1204, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, Massachusetts.

Connell, J. (1990), *Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot*, Academic Press, Cambridge, Massachusetts. also MIT TR-1151.

Cooper, R., Shallice, T. & Farringdon, J. (1995), Symbolic and continuous processes in the automatic selection of actions, *in* J. Hallam, ed., 'Hybrid Problems, Hybrid Solutions, Frontiers in Artificial Intelligence and Applications', IOS Press, Amsterdam, pp. 27–37.

Correia, L. & Steiger-Garção, A. (1995), A Useful Autonomous Vehicle With a Hierarchical Behavior Control, *in* F. Moran, A. Moreno, J. Merelo & P. Chacon, eds, 'Advances in Artificial Life (Third European Conference on Artificial Life)', Springer-Verlag, Berlin, Germany, pp. 625–639.

Dreyfus, H. L. (1992), *What Computers Still Can't Do*, MIT Press, Cambridge, MA.

Gallistel, C., Brown, A. L., Carey, S., Gelman, R. & Keil, F. C. (1991), Lessons From Animal Learning for the Study of Cognitive Development, *in* S. Carey & R. Gelman, eds, 'The Epigenesis of Mind', Lawrence Erlbaum, Hillsdale, NJ.

Garcia, J. & Koelling, R. A. (1966), 'The relation of cue to consequence in avoidance learning', *Psychonomic Science* **4**, 123–124.

Henson, R. N. A. (1996), Short-term Memory for Serial Order, PhD thesis, St. John's College.

Hexmoor, H. H. (1995), Representing and Learning Routine Activities, PhD thesis, State University of New York at Buffalo.

Hineline, P. & Rachlin, H. (1969), 'Escape and avoidance of shock by pigeons pecking a key', *Journal of Experimental Analysis of Behavior* **12**, 533–538.

Hinton, G. E. & Nowlan, S. J. (1987), 'How Learning Can Guide Evolution', *Complex Systems* **1**, 495–502.

Horswill, I. (1997), 'Visual architecture and cognitive architecture', *Journal of Experimental & Theoretical Artificial Intelligence* **9**(2/3), 277–293.

Horswill, I. D. (1993), Specialization of Perceptual Processes, PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Horswill, I. D. (1995), Visual routines and visual search, *in* 'Proceedings of the 14th International Joint Conference on Artificial Intelligence', Montreal.

Kaelbling, L. P. (1997), Why Robbie Can't Learn: The Difficulty of Learning in Autonomous Agents, *in* 'Proceedings of the 15th International Joint Conference on Artificial Intelligence', Morgan Kaufmann, Nagoya. IJCAI Computers and Thought Award talk.

Karmiloff-Smith, A. (1992), *Beyond Modularity*, MIT Press, Cambridge, MA.

Levison, L. (1996), Connecting Planning and Action via Object-Specific Reasoning, PhD thesis, School of Engineering and Applied Science, Computer and Information Science Department.

Lorenz, K. (1973), *Foundations of Ethology*, Springer-Verlag, New York.

Maes, P. (1989), How To Do the Right Thing, A.I. Memo 1180, MIT, Cambridge, MA.

Maes, P. (1990), *Designing Autonomous Agents : Theory and Practice from Biology to Engineering and back*, MIT Press, Cambridge, MA.

Maes, P., Matarić, M. J., Meyer, J.-A., Pollack, J. & Wilson, S. W., eds (1996), *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, MIT Press, Cambridge, MA.

Malcolm, C. (1992), Personal communication.

Malcolm, C. & Smithers, T. (1990), Symbol Grounding via a Hybrid Architecture in an Autonomous Assembly System, *in* P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', MIT Press, Cambridge, MA, pp. 123–144.

Matarić, M. J. (1997), 'Behaviour-based control: examples from navigation, learning, and group behaviour', *Journal of Experimental & Theoretical Artificial Intelligence* **9**(2/3), 323–336.

McGonigle, B. (1991), Incrementing Intelligent Systems by Design, *in* J.-A. Meyer & S. Wilson, eds, 'From Animals to Animats', MIT Press, Cambridge, MA, pp. 478–485.

McGonigle, B. & Chalmers, M. (forthcoming), *The Growth of Intelligence in Complex Systems*, MIT Press, Cambridge, MA.

Minsky, M. (1986), *The Society of Mind*, Simon and Schuster, New York, New York.

Nilsson, N. (1984), Shakey the Robot, Technical note 323, SRI International, Menlo Park, California.

Norman, D. A. & Shallice, T. (1986), Attention to Action: Willed and Automatic Control of Behavior, *in* R.Davidson, G. Schwartz & D. Shapiro, eds, 'Consciousness and Self Regulation: Advances in Research and Theory', Vol. 4, Plenum, New York, pp. 1–18.

Rhodes, B. (1996), PHISH-Nets: Planning Heuristically in Situated Hybrid Networks, Master's thesis, MIT Media Lab.

Seyfarth, R. M., Cheney, D. L. & Marler, P. (1980), 'Monkey responses to three different alarm calls: Evidence of predator classification and semantic communication', *Science* **14**, 801–803.

Steels, L. (1994), Building Agents with Autonomous Behavior Systems, *in* L. Steels & R. Brooks, eds, 'The 'artificial life' route to 'artificial intelligence'. Building situated embodied agents.', Lawrence Erlbaum Associates, New Haven.

Sundareswaran, V. & Vaina, L. M. (1994), Learning Direction in Global Motion: Two Classes of Psychophysically-Motivated Models, *in* G. Tesauro, D. Touretzky & T. Leen, eds, 'Advances in Neural Information Processing Systems 7', Denver, CO.

Tanji, J. (1996), Involvement of motor areas in the medial frontal cortex of primates in temporal sequencing of multiple movements, *in* R. Caminiti, K. Hoffmann, F. Lacquaniti & J. Altman, eds, 'Vision and Movement: Mechanisms in the Cerebral Cortex', Vol. 2, Human Frontier Science Program, Strasbourg, pp. 126–133.

Ullman, S. (1984), 'Visual Routines', *Cognition* **18**, 97–159.

Whitehead, S. D. (1992), Reinforcement Learning for the Adaptive Control of Perception and Action, Technical Report 406, University of Rochester Computer Science, Rochester, NY.

Wilson, M. & McNaughton, B. (1994), 'Reactivation of Hippocampal Ensemble Memories During Sleep', *Science* **261**, 1227–1232.

Wilson, M. S. (1996), 'Reliability and Flexibility — A Mutually Exclusive Proble m for Robotic Assembly?', *IEEE Transactions on Robotics and Automation.*

Winston, P. H. (1992), *Artificial Intelligence*, third edition edn, Addison-Wesley, Reading, Massachusetts.

Wooldridge, M. & Jennings, N. R. (1995), 'Intelligent Agents: Theory and Practice', *Knowledge Engineering Review.*