Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior

Maja J Matarić
Volen Center for Complex Systems
Computer Science Department
Brandeis University
Waltham, MA 02254
tel: (617) 736-2708, fax: (617) 736-2741
maja@cs.brandeis.edu

Abstract

This paper describes the main properties of behavior-based approaches to control. Different approaches to designing and using behaviors as basic units for control, representation, and learning are illustrated on three empirical examples of robots performing navigation and path-finding, group behaviors, and learning behavior selection.

1 Introduction

An architecture provides a set of principles for organizing control systems. In addition to supplying structure, it imposes constraints on the way control problems can be solved. In this paper we explore the constraints of *behavior-based* approaches to control, and demonstrate them on three architectures that were used to implement robots that successfully performed navigation and path-finding, group behaviors, and learning of behavior selection. In each case, we focus on the different ways behaviors are defined, modularized, and combined.

This paper is organized as follows. Section 2 gives an overview of basic approaches to autonomous agent control: reactive, planner-based, and hybrid. Section 3 presents the defining properties of behavior-based systems. Section 4 discusses the capability tradeoffs of the four control approaches. Section 5 describes and discusses the three experimental examples of behavior-based systems. Section 6 discusses some of the key issues in control, including centralized versus decentralized solutions, and extensions to learning. Section 7 summarizes and concludes the paper.

2 Deliberative and Reactive Approaches

At one extreme of the agent control spectrum lie traditional top-down *planner-based* or *deliberative* strategies that typically rely on a centralized world model for verifying sensory information and generating actions in the world (Giralt, Chatila & Vaisset 1983, Chatila

& Laumond 1985, Moravec & Cho 1989, Laird & Rosenbloom 1990). The information in the world model is used by the planner to produce the most appropriate sequence of actions for the agent. These approaches allow for explicitly formulating the task and goals of the system, and estimating the quality of the agent's performance. However, uncertainty in sensing and action, and changes in the environment, can require frequent replanning the cost of which may be prohibitive for complex systems. Planner-based approaches have been criticized for scaling poorly with the complexity of real-world problems, and making real-time reaction to sudden world changes impossible (Brooks 1990, Brooks 1991*b*).

Various approaches for achieving real-time performance in autonomous agents have been proposed. *Purely reactive* bottom-up approaches are featured in various implemented systems. They embed the agent's control strategy into a collection of preprogrammed condition-action pairs with minimal internal state (Brooks & Connell 1986, Agre & Chapman 1987, Connell 1990). Reactive systems maintain no internal models and perform no search. Typically, they apply a simple functional mapping between stimuli and appropriate responses, usually in the form of a look-up, whether it be in a table, a set of reactive rules, a simple circuit, a vector field, or a connectionist network. All of those implementations are variations of the same theme of constant-time run-time direct encodings of the appropriate action for each input state. However encoded, these mappings rely on a direct coupling between sensing and action, and fast feedback from the environment. Purely reactive strategies have proven effective for a variety of problems that can be completely specified at design-time. However, such strategies are inflexible at run-time due to their inability to store information dynamically.

The division between reactive and deliberative strategies can be drawn based on the type and amount of computation performed at run-time. Reactive, constant-time run-time strategies can be derived from a planner, by computing all possible plans off-line beforehand. For example, situated automata achieve real-time performance by compiling all of the system's goals and associated plans of achievement into a language that compiles into circuits that have constant-time computation properties (Rosenschein & Kaelbling 1986). In general, the entire control system of an agent can be precompiled as a decision graph into a collection of reactive rules ("a universal plan") (Schoppers 1987). While theoretically appealing, these strategies often scale poorly with the complexity of the environment and the agent's control system.

Hybrid architectures attempt a compromise between purely reactive and deliberative approaches, usually by employing a reactive system for low-level control and a planner for higher-level decision making. Hybrid systems span a large and diverse body of research. It includes *reactive planning or reactive execution* used in Reactive Action Packages (RAPs), higher-level primitives for planning that hide and take care of the details of execution (Firby 1987), PRS (Procedural Reasoning System), an architecture for flexible control rule invocation (Georgeff & Lansky 1987), Schemas (Arkin 1989), Internalized Plans (Payton 1990), Contingency Plans (Connell 1991), and others. Hybrid solutions tend to separate the control system into two or more communicating but otherwise independent parts. In most cases, the low-level reactive process takes care of the immediate safety of the agent, while the higher level uses the planner to select action sequences.

3 Behavior-Based Approaches

Behavior-based approaches are an extension of reactive architectures and also fall between purely reactive and planner-based extremes (Brooks 1986, Maes 1989). Although often conflated in the literature, reactive and behavior-based systems are fundamentally different. While behavior-based systems embody some of the properties of reactive systems, and usually contain reactive components, their computation is not limited to look-up and execution of simple functional mappings. As we will demonstrate in the experimental section of the paper, behaviors can be employed to store various forms of state and implement various types of representations. This is one of the main reasons behind the difficult and persisting problem of defining the meaning of "behavior" in behavior-based systems. Different authors use varying, and sometimes even contradictory, meanings of behavior. In this paper, we will define the specific meaning of behavior our work has employed in each of the implementations, as well as outline some general constraints on behavior design.

Behavior-based strategies have been governed by a general set of constraints that allow for a certain freedom of interpretation. This freedom has been interpreted as a lack of structure by some critics and has been a source of annoyance to some designers. It has also generated complaints about a lack of rigorous definitions and associated system analysis, since the more interpretative freedom there is, the more difficult it is to perform comparative evaluation across different systems. However, this freedom of interpretation has also encouraged various novel and efficient ideas to be proposed and implemented in using the behavior-based paradigm.

In general, behavior-based systems do not employ centralized representations operated on by one or more reasoning engines. Instead, they typically rely on various forms of distributed representations and perform distributed computations on them. In some cases, the representations are not static, manipulable structures, but active, procedural processes (Matarić 1990*a*, Brooks 1991*a*).

Furthermore, behaviors are typically more time-extended than actions of reactive systems. The latter often produce coherent externally measurable output behaviors from the interaction of their rules in a particular environment. These are commonly labeled "emergent" in that no place in the system explicitly specifies them. In contrast, behavior-based systems often internally specify such behaviors. Their "emergent properties" (Steels 1994) then result from the interaction of the behaviors and the world, are are typically higher-level. Section 4.1 describes a specific example of a robot navigation system that employs both reactive rules and behaviors. The two are compared, in order to clarify this subtle and rather confusing distinction.

The organizational methodology of behavior-based systems differs from the classical hierarchical systems in its approach to modularity. The philosophy mandates that behavior execution not be simply serialized, thus reducing the system to one that could be implemented through more traditional centralized means. The organizational methodology concerns the coordination of a multitude of behaviors, thus making behavior-arbitration one of the central design challenges of such systems. For the sake of simplicity, in the majority of systems the solution is a built-in, fixed control hierarchy imposing a priority ordering on the behaviors, much like such hierarchies have been used to employ priority schemes over reactive rules, such as for example in the Subsumption

Architecture (Brooks 1986). More flexible, although often less tractable, solutions have been suggested, commonly based on selecting an output behavior by computing a multivariable function implicitly encoded in behavior activation levels, such as voting schemes (Payton, Keirsey, Kimble, Krozel & Rosenblatt 1992) and spreading of activation (Maes 1991).

In summary, the general constraints on behavior-based systems roughly mandate that behaviors be relatively simple, incrementally added to the system, that their execution not be serialized, that they be more time-extended than simple atomic actions of the particular agent, and that they interact with other behaviors through the world rather than internally through the system. Before giving examples to illustrate the specifics of applying these constraints and principles in practice, we briefly discuss the tradeoffs in capability across the different approaches to control we have discussed: deliberative, reactive, hybrid, and behavior-based.

4 Capability Tradeoffs

Purely reactive systems have historically been assumed to be less powerful than their behavior-based and planner-based counterparts. However, it has been shown that for any well-defined task, well-known environment, and well-equipped robot, purely reactive solutions can be used to solve complex control problems (Schoppers 1987, Chapman 1989). The burden on the designer is to predict and elaborate all possible relevant input states, and the appropriate corresponding output actions. Because of their minimal computational overhead, purely reactive approaches achieve great run-time efficiency but their limited representational power results in a lack of run-time flexibility. The reactive versus deliberative dilemma is best reflected in the tradeoff between the amount of built-in information and the amount of on-line computation. Depending on the task, it may be easier to hardwire the action rules, or it may be simpler to maintain an internal world model.

While the drawbacks of pure reactivity are obvious, the tradeoffs between behavior-based and planner-based systems are less clear. In the end, the power, elegance, and complexity of behavior-based systems all stem from the particular way in which they define and apply behaviors, their basic unit of abstraction and control. The next section illustrates three implemented robot systems using variations on the behavior-based theme applied to the tasks of navigation and path-finding, group behavior, and learning.

5 Experimental Examples

This section describes three implemented behavior-based robotic systems, each illustrating a different aspect of the control approach. The first, Toto, demonstrates the mix of reactive rules and different types of behaviors to implement deliberation and optimization capabilities in a behavior-based system. The second, Nerd Herd, illustrates the use of basis behaviors as foundational primitives for generating low-level and high-level tasks in multi-robot control. The third, Don Group, demonstrates how behavior-based control can be used as an effective substrate for learning. The three are illustrative of some possible variations on the common theme of designing behaviors as basic units for control, representation, and learning in real-time, embodied, situated systems.

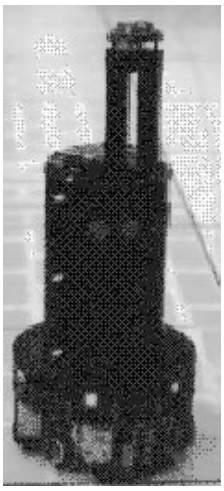

Figure 1: The mobile robot used in the navigation and mapping experiments, equipped with a ring of 12 sonars and a compass.

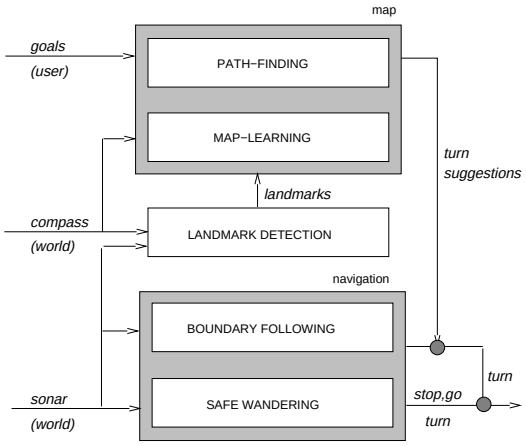

Figure 2: The control architecture for Toto, a behavior-based robot that performed navigation, boundary following, landmark detection, map learning, and path-finding.

5.1 Navigation and Path-Finding

The capability for constructing and updating internal representations of the world at runtime does not exist in reactive systems. It was assumed that the same was the case for behavior-based systems as well until Matarić (1990*a*) described a behavior-based robot, Toto, equipped with a ring of sonars and a compass (Figure 1), that was built with the goal of demonstrating both higher-level reasoning and robust real-time reaction in a non-hybrid behavior-based system. Toto demonstrates real-time obstacle avoidance, boundary following, landmark detection, map construction, and path finding. As shown in Figure 2, Toto's general capabilities appear typical of a deliberative system, but were implemented with a behavior-based approach Matarić (1992).

Toto's control system employs an interaction of reactive rules and behaviors. Navigation and boundary tracing result from the interaction of several simple reactive rules that move and turn the robot. These take inputs from the sonar, compass, and motor current sensors, and send outputs directly to the motors. In contrast, landmark detection is achieved with four behaviors, each of which is a perceptual filter that monitors the exter-

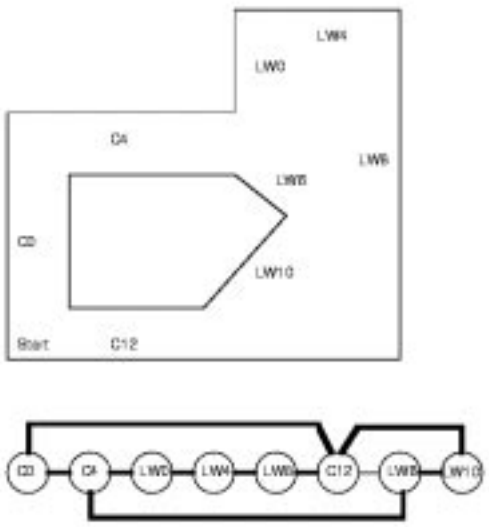

Figure 3: The behaviors in the map correspond to particular landmarks in the environment (e.g. LW0 means "left wall at compass bearing 0"). They are connected with topological and communication links.

nal world (through the sonars and the compass) and the robot's movement (through the base). These behaviors have no direct effect on the robot's motion, but monotonically increase and decrease activation levels related to particular landmark types they are tuned to recognize. For example, a "corridor-finding" behavior observes the movement of the robot and its lateral sensory readings in order to continuously update its confidence level if the robots is moving straight and sensing boundaries on both sides over an extended time period. When the activation level of any landmark reaches a pre-specified threshold, the behavior sends a message to the map behaviors.

The map behaviors are a collection of initially empty behavior-shells that get assigned to specific landmarks and their properties as they are discovered. For example, a map behavior may represent a corridor, at compass bearing 8, with an approximate length of 5 units. These map behaviors are attached to each other with links that serve both as topological and communication connections (Figure 3).

The landmark detector behaviors broadcast their outputs to all of the map behaviors. The map behavior that best matches the received landmark becomes active thus effectively localizing the robot within the map. If no landmark matches, a new one is added to the map by assigning the landmark properties to an empty behavior-shell. The topological connections in the map, augmented with approximate physical lengths of the landmarks, allow for both topological and metric path optimization.

In Toto, path planning is achieved by spreading activation through the map (i.e., the behavior network), from the selected goal behavior through all the nodes in the map, until the currently active map behavior is reached. The activation is propagated by each behavior that receives on to its neighbor(s) as a message. If the message contains the accumulated length of all bypassed landmark behaviors the shortest resulting path can can be chosen. When activation reaches the current behavior (i.e., the node representing the robot's position in the map) it generates a motion command to the wheels of the robot, turning and moving it in the direction of the next map landmark/behavior on the shortest path to the goal. The activation continues to spread throughout the map until the robot arrives at the goal, i.e., until its current and goal landmarks are identical. However, if the optimal path to the goal is blocked, the robot eventually attempts an alternative path, and the blocked topological link is removed.

Toto's behavior arbitration is simple in that all of its navigation rules and landmark detec-

tion behaviors are active in parallel and do not interfere with one another. Obstacle avoidance has the highest precedence and halts all others that send motor outputs. Unlike the rest of the system, map behaviors laterally inhibit each other so only one is active at a time, based on localization information from the sensors and a small amount of history built into the activation spreading (Matarić 1990*a*).[1]

Details of Toto's implementation and evaluation can be found in Matarić (1994*a*). Toto is a demonstration of a traditional planning task implemented with a behavior-based system using an *active representation*, i.e., one that is procedural and distributed. At a high level of description, distributed spreading of activation used in Toto's map representation can be viewed as a simple distributed search mechanism, but the semantics of the behaviors in Toto's system are more complex and significant than those of simple graph nodes. Furthermore, the use of reactive rules and behaviors throughout the system, from the low-level navigation and control to the map representation, is a departure from both the planner-based and hybrid approaches to similar problems.

5.2 Multi-Agent Control

The challenge of scaling up is commonly posed to behavior-based systems. We have explored scaling up not only in terms of the number of behaviors within a single agent (as shown in Toto), but also in the number of interacting agents. Extending the planning paradigm[2] from single-agent to multi-agent domains requires expanding the global state space to include the state of each of the agents. Such a global state space G is exponential in the number of agents: $|G| = s^a$ where s is the size of the state space of each agent, here assumed to be equal for all agents, or at worst the maximum for all agents, and a is the number of agents. This makes the problem of online planning intractable for all but the smallest group sizes. Furthermore, since global planning requires communication between the agents and the controller, the bandwidth can grow with the number of agents. Additionally, the uncertainty in perceiving state grows with the increased complexity of the environment. All of these properties conspire against global planner-based approaches for problems involving multiple agents acting in real-time in dynamic, noisy environments.

At the other end of the control spectrum, extending the reactive and behavior-based approaches to multi-agent domains results in completely distributed systems with no centralized controller at any level. The systems are identical at the local and global levels: at the global level the systems are collections of reactive agents each executing task-related rules or behaviors relying only on local sensing and communication. Since all control in such distributed systems is local, it scales well with the number of agents, does not require global communication, and is more robust to sensor and effector errors. However, global consequences of local interactions between agents are difficult to predict.

Our work, described in Matarić (1994*a*), introduces an approach to structuring local reactive rules and behaviors into a parsimonious set to be used as a basis for programming a collection of robots in a coherent, scalable

[1] Inspired by the neurophysiology of rat navigation, Toto's behaviors bear a loose resemblance to some theories of the organization of the rat hippocampus (Matarić 1990*b*).

[2] The planning paradigm includes traditional and hybrid systems. In terms of multi-agent extensions, hybrid systems fit into the planner-based category since their collective behavior is generally a result of a plan produced by a global controller.

Figure 4: The family of 20 ISX mobile robots used in the group behavior experiments. The robots are equipped with IRs, contact sensors, grippers, position sensors, and radio communication.

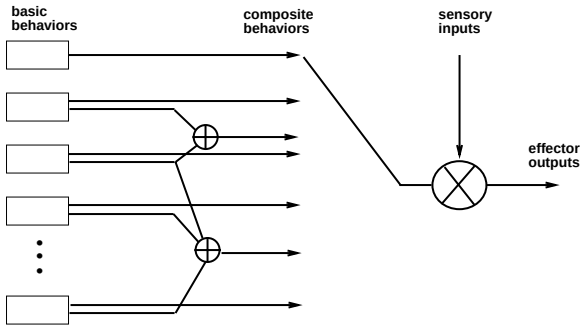

Figure 5: The control architecture for generating group behaviors consists of direct and temporal combinations (i.e. summing behaviors and switching between them) of subsets from a fixed basis behavior set. Direct combinations or sums are indicated with $\oplus$, and temporal combinations or switches with $\otimes$.

fashion. In this work we use a specific definition of behavior: a control law that satisfies a set of constraints to achieve and maintain a particular goal. For example, *safe-wandering* is a behavior that maintains the goal of avoiding collisions while the agent is moving. Behaviors are designed to utilize the interaction dynamics between the agents (Agre 1988); they combine constraints from the agent, such as its mechanical characteristics, and the constraints for the environment, such as the sensory information the agent can obtain.

We select a special set of such behaviors, called *basis behaviors*, that are not further reducible to each other, and can be combined to generate a large repertoire of higher-level behaviors. We dually constrain the process of choosing the set of basis behaviors for a given domain: from the bottom-up by the dynamics of the agent and the environment, and from the top-down by the agent's goals as specified by the task (Matarić 1994*a*). The combination of the two types of constraints helps the designer prune the agent's behavior space and find an efficient basis set.

The basis behavior methodology was demonstrated on the Nerd Herd, a collection of 20 ISX mobile robots equipped with IRs, contact sensors, grippers, position sensors, and radio communication (Figure 4). The behavior set chosen for spatial interactions among the robot herd consists of: safe-wandering, following, aggregation, dispersion, and homing. The algorithms for the behaviors were tested relative to following fixed criteria: repeatability, stability, robustness, and scalability. They were also compared to hierarchical and centralized alternatives.

Basis behaviors are intended as building blocks for generating higher-level behaviors for performing various tasks. The behaviors are embedded in an architecture that allows two types of combination operators: direct combination by summation ($\oplus$), and temporal combination by switching ($\otimes$) (Figure 5). Summing adds the contributions/outputs from multiple behaviors into a single output to the relevant effector (in our case the wheels of the robot). For example, we implemented flocking as a result of summing the outputs of safe-wandering, aggrega-

tion and dispersion. If the flock is expected to go toward a particular location/direction, then the outputs of homing is also included. Since all of the basis behaviors output velocity vectors, these are fully compatible and summable.

Switching behaviors applies lateral inhibition to all but one of the basis behaviors at a time, much like the mechanism employed in Toto's map network described in the previous section. Only one behavior has complete control of the relevant effector. For example, we implemented foraging as a result of activating safe-wondering when a robot needed a puck, homing when it had a puck, dispersion when crowded with other agents, following when near another robot with the same state[3], and so on. Foraging results from activating appropriate behaviors under the right conditions.

Details of the Nerd Herd experiments and evaluation can be found in Matarić (1994*a*). The experiments with basis behaviors demonstrate an approach toward principled, parsimonious development of basis modules for a behavior-based system. The need for a uniform, consistent, and robust unit of control is particularly important in the multi-robot environment where behavior interference and the need for arbitration are amplified. The Nerd Herd experiments propose a pragmatic notion of behavior design and modularization, which can be successfully extended to learning, as described next.

5.3 Learning Behavior Selection

Basis behaviors were introduced as a methodology for structuring simple rules into flexible and effective higher-level behavior repertoires. In addition to being appropriate units for control, such behaviors also serve as an efficient substrate for enabling learning of behavior selection. In the set of experiments described in this sections, the robots learn what behaviors to activate and which ones to inhibit in order to forage effectively in a group. They learn the foraging controller, much like the handcrafted one described above, from the information and feedback (i.e., reinforcement) they receive through their interaction with each other and the environment.

Figure 6: The family of four IS Robotics mobile robots used in the group behavior experiments. The robots are equipped with IRs, contact sensors, grippers, position sensors, and radio communication.

The traditional formulation of reinforcement learning, currently the most popular learning paradigm for autonomous agents, is posed in terms of states, actions, and reinforcement. The agent is at all times in one of a finite number of possible states, it takes an action, and may or may not receive some reinforcement. Over time, it learns to correlate states and actions in order to maximize reinforcement.

Typical representations of states and ac-

[3] A robot with a puck will follow another with a puck, since in our system they are going on the unique home location. A robot without a puck will avoid other robots.

tions have such a low representational granularity that they result in prohibitively slow learning for physical robots. In Matarić (1994*b*), we demonstrated a simple reformulation of reinforcement learning based on behaviors (using the same formulation as basis behaviors above), instead of actions, as the basic representational units. The use of such behaviors, which achieve goals but hide low-level details of control, allows for replacing the complete state space with a much smaller set of *conditions*, the necessary and sufficient subsets of state required for triggering the behavior set, i.e., the preconditions for each of the behaviors. Since conditions are many fewer than states, this greatly diminishes the agent's learning space (i.e., the search space of the learning system), and speeds up any learning algorithm that is now applied to this representation. In our experiments we implemented a simple learning algorithm that accumulated received reinforcement over time in order to correlate conditions with behaviors. We compared the performance of this algorithm to the standard Q-learning approach, as described below.

The learning problem we implemented, choosing the best (highest reinforcement) behavior for each condition, can also be viewed as learning the preconditions for each behavior. We demonstrated it on a group of four IS Robotics R2 mobile robots dubbed the Don Group[4] learning to foraga (Figure 6), quipped with the same sensors as the Nerd Herd, and using the same basis behavior set. In order to make learning feasible in such a noisy, dynamic group scenario, it was necessary to use more than just delayed reinforcement such as a reward when a robot dropped a puck in the home region. Given the non-stationary nature of the environment, it was necessary either to build-in much of the solution, or to allow the robots to receive richer feedback from the world as they explored.

We introduced *shaped reinforcement* in two forms: feedback after the completion of a time-extended behavior, and feedback during the execution of a time-extended behavior, in each case if any is available. Providing feedback at the completion of a behavior helps the robot correlate conditions and behaviors thus learning when to execute any given behavior. This feedback combined information multi-modally from all of the sensors (e.g., IRs and positioning sensors and contact sensors and internal progress estimators) as well as from intermittent reinforcement generated by various events in the environment (e.g., finding a puck, getting home, dropping a puck).

Providing some feedback during the execution of a behavior helps the robot explore the behavior space more effectively by having a meaningful metric of when to continue and when to terminate a particular behavior. This feedback comes from so-called *internal progress estimators* that are naturally available for some of the robot's behaviors (e.g., approaching home, being crowded by other agents, etc.), while not for others (e.g., searching for pucks). These progress estimators are generic and apply to more than one learning task.

In our system, behaviors are triggered and terminated by events, which can be external (e.g., being bumped by another robot, dropping a puck, etc.) or internal (e.g., low battery power, lack of progress, etc.). Event-driven behavior termination is more natural than the use of an arbitrary time period. Since behaviors achieve goals, they take varying amounts of time to execute depending on the particular situation. As the situation changes dynamically with the movement of the entire group, it is not realistic to use arbitrary behavior termi-

[4]Don Quixote, Don Corleone, Donna E. Mobile, and the Donald. Don' Work did not participate.

nation and switching methods. Event-based behavior switching benefits from progress estimators since it prevents the robot from getting stuck in a behavior and from oscillating between behaviors.

As in the case of hand-crafted foraging, only one of the basis behaviors is active at a time, and the robots use the reinforcement to learn the switching circuit. The reinforcement functions described above are a form shaping the robot's behavior toward the desired efficient foraging without building-in the solution or providing supervised reinforcement. In our case, all reinforcement was internally generated, through the robot's own reward and punishment system, itself implemented in the form of behaviors. These internal reinforcement behaviors are similar to Toto's landmark detectors. Multi-model feedback behaviors are perceptual filters that monitor the environment, detect particular events, and deliver appropriate reinforcement. Similarly, progress estimator behaviors are even more akin to landmark detectors in that they also utilize activation levels, in this case representing monotonically increasing boredom that, when it reaches a pre-set threshold, triggers the termination of the currently executing, unproductive behavior, and selects another.

The described condition and behavior representation and the shaped reinforcement were tested on groups of 3 and 4 robots learning to forage. The robots were successful in automatically acquiring efficient group foraging within less than 15 minutes per trial. The complete learned policy, fully described in Matarić (1994*a*), consisted of a mapping between the power set of the conditions and the behavior set. We compared the performance of our approach, using multi-modal feedback and progress estimators to two alternatives: 1) just the use of multi-modal feedback and no progress estimators, and 2) the use of Q-learning. While our complete approach learned foraging in over 95% of the experiments (over 20 trials with two different group sizes), the approach using only multi-modal feedback, i.e., reinforcement only at the termination of a behavior, only succeeded in learning about 60% of the foraging policy. Even more interestingly, Q-learning could only acquire about 30% of the correct policy, due to the lack of shaped reinforcement. Q is very effective at learning in non-dynamic domains where reinforcement can be propagated back in time. However, our noisy, non-stationary, and complex domain required shaped reinforcement in order to both enable learning and make it efficient (Matarić 1994*b*).

The details of the learning experiments and their evaluation can be found in Matarić (1994*a*). The experiments demonstrate basis behaviors as an effective substrate for automated synthesis of new higher-level behaviors. The results begin to address the on-going challenge of automatic synthesis of behavior-based systems which are often accused of being difficult to program by hand. We are currently exploring the possibility of automating the design of the basis behaviors themselves through the use of genetic learning techniques applied to interleaved on-line and off-line trials. The general problem of automated synthesis of robot controllers has barely begun to be addressed, and scaling issues of the existing techniques are not encouraging (Matarić & Cliff 1996).

6 Discussion

One of the main goals of behavior-based systems has been to demonstrate that distributed approaches to autonomous agent control are feasible, efficient, and robust. In all cases we described, centralized behavior coordination was shown to be unnecessary. Toto demon-

strated that it could be effectively bypassed in a traditionally centralized task, while the Nerd Herd showed that decentralized control is more effective and scales better for large group sizes.

The real-time capabilities of well-designed behavior-based systems should guarantee timely achievement and maintenance of all goals. While not all reactive and behavior-based systems employ consistent rule and behavior design, our work has demonstrated that principled methods can be applied to both behavior design and evaluation. We have briefly mentioned some of the performance metrics used in evaluating the Nerd Herd group behaviors we described, based on *a priori* formal evaluation criteria. We also proposed similar formal criteria for designing and selecting basis behaviors in Matarić (1994*a*).

Extending control systems to include learning capabilities is becoming an increasingly more important issue in autonomous agent control. The learning system we demonstrated is a direct and natural extension of the basis behavior approach using a simple switching arbitration mechanism. We are optimistic that learning can be similarly applied at multiple levels of a behavior-based system: at the evolutionary scale for automatically generating basis behaviors, at the behavior level for tuning parameters and functions learning, and at the behavior combination and selection level, as demonstrated with the Don Group. Behaviors have been shown to be a good basis for learning in other robotics tasks; specifically, Mahadevan & Connell (1991) used them in applying reinforcement learning to a box-pushing task, and Maes & Brooks (1990) used them for learning to walk on a six-legged mobile robot. We have subsequently employed the same methodology as the one we applied to the Don Group to learn a more tightly-coupled coordination behavior. In it, two six-legged robots learned to push a box to a light and to communicate to each other what actions to take, since neither individually had sufficient sensors or effectors to complete the task (Simsarian & Matarić 1995, Matarić 1996). The results of both sets of learning experiments provide further encouragement for continuing to work with and scale up behavior-based systems.

7 Conclusions

The control architecture constrains the way an autonomous agent senses, reasons, and acts. We have described the ways behavior-based control guides and constrains agent design, and discussed its properties relative to some standard alternatives. The design of *behaviors*, the basic unit of behavior-based controllers, determines the effectiveness of the control system, and is the main challenge of the approach, much like abstraction and representation are among the grand challenges of AI. To illustrate the versatility and flexibility of behaviors, we described three different implemented applications of the notion and approach: navigation and path-finding, group behavior, and learning behavior selection. For each example, we discussed the definition of behaviors, their use, and their associated arbitration mechanism.

Much work remains to be done in exploring the potential of behavior-based systems and comparing their performance to other alternatives. At the current stage of development in robotics and autonomous agent control, it will be most productive to gain as much principled empirical evidence as possible, in order to learn more about what truly complex tasks and environments demand.

Acknowledgements

The research reported here was done partly at the MIT Artificial Intelligence Laboratory, and subsequently continued at the author's Interaction Laboratory at the Brandeis University Volen Center for Complex Systems and the Computer Science Department. The MIT work was supported in part by the Jet Propulsion Laboratory and in part by the Advanced Research Projects Agency under the Office of Naval Research. The work at Brandeis is supported by the Office of Naval Research Grant N00014-95-1-0759 and the National Science Foundation Infrastructure Grant CDA-9512448. The author wishes to thank the two anonymous reviewers for their detailed constructive comments.

References

Agre, P. E. (1988), The Dynamic Structure of Everyday Life, PhD thesis, MIT.

Agre, P. E. & Chapman, D. (1987), Pengi: An Implementation of a Theory of Activity, *in* 'Proceedings, AAAI-87', Seattle, WA, pp. 268–272.

Arkin, R. C. (1989), Towards the Unification of Navigational Planning and Reactive Control, *in* 'AAAI Spring Symposium on Robot Navigation', pp. 1–5.

Brooks, R. A. (1986), 'A Robust Layered Control System for a Mobile Robot', *IEEE Journal of Robotics and Automation* **RA-2**, 14–23.

Brooks, R. A. (1990), Elephants Don't Play Chess, *in* P. Maes, ed., 'Designing Autonomous Agents', MIT Press, pp. 3–15.

Brooks, R. A. (1991*a*), Intelligence Without Reason, *in* 'Proceedings, IJCAI-91'.

Brooks, R. A. (1991*b*), 'Intelligence Without Representation', *Artificial Intelligence* **47**, 139–160.

Brooks, R. A. & Connell, J. H. (1986), Asynchronous Distributed Control System for a Mobile Robot, *in* 'SPIE', Cambridge, MA.

Chapman, D. (1989), 'Penguins Can Make Cake', *AI Magazine* **10**(4), 45–50.

Chatila, R. & Laumond, J.-C. (1985), Position Referencing and Consistent World Modeling for Mobile Robots, *in* 'IEEE International Conference on Robotics and Automation'.

Connell, J. H. (1990), *Minimalist Mobile Robotics: A Colony Architecture for an Artificial Creature*, Academic Press.

Connell, J. H. (1991), SSS: A Hybrid Architecture Applied to Robot Navigation, *in* 'IEEE International Conference on Robotics and Automation', Nice, France, pp. 2719–2724.

Firby, R. J. (1987), An investigation into reactive planning in complex domains, *in* 'Proceedings, Sixth National Conference on Artificial Intelligence', Seattle, pp. 202–206.

Georgeff, M. P. & Lansky, A. L. (1987), Reactive Reasoning and Planning, *in* 'Proceedings, Sixth National Conference on Artificial Intelligence', Seattle, pp. 677–682.

Giralt, G., Chatila, R. & Vaisset, M. (1983), An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots, *in* M. Brady & R. Paul, eds, 'First International Symposium on Robotics Research', MIT Press, Cambridge, MA.

Laird, J. E. & Rosenbloom, P. S. (1990), Integrating, Execution, Planning, and Learning in Soar for External Environments, *in* 'Proceedings, AAAI-90', pp. 1022–1029.

Maes, P. (1989), The Dynamics of Action Selection, *in* 'IJCAI-89', Detroit, MI, pp. 991–997.

Maes, P. (1991), Learning Behavior Networks from Experience, *in* F. Varela & P. Bourgine, eds, 'Toward A Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life', MIT Press, pp. 48–57.

Maes, P. & Brooks, R. A. (1990), Learning to Coordinate Behaviors, *in* 'Proceedings, AAAI-91', Boston, MA, pp. 796–802.

Mahadevan, S. & Connell, J. (1991), Automatic Programming of Behavior-based Robots using Reinforcement Learning, *in* 'Proceedings, AAAI-91', Pittsburgh, PA, pp. 8–14.

Matarić, M. J. (1990*a*), A Distributed Model for Mobile Robot Environment-Learning and Navigation, Technical Report AI-TR-1228, MIT Artificial Intelligence Laboratory.

Matarić, M. J. (1990*b*), Navigating With a Rat Brain: A Neurobiologically-Inspired Model for Robot Spatial Representation, *in* J.-A. Meyer & S. Wilson, eds, 'From Animals to Animats: International Conference on Simulation of Adaptive Behavior', MIT Press, pp. 169–175.

Matarić, M. J. (1992), 'Integration of Representation Into Goal-Driven Behavior-Based Robots', *IEEE Transactions on Robotics and Automation* **8**(3), 304–312.

Matarić, M. J. (1994*a*), Interaction and Intelligent Behavior, Technical Report AI-TR-1495, MIT Artificial Intelligence Lab.

Matarić, M. J. (1994*b*), Reward Functions for Accelerated Learning, *in* W. W. Cohen & H. Hirsh, eds, 'Proceedings of the Eleventh International Conference on Machine Learning (ML-94)', Morgan Kauffman Publishers, Inc., New Brunswick, NJ, pp. 181–189.

Matarić, M. J. (1996), Learning in Multi–Robot Systems, *in* G. Weiss & S. Sen, eds, 'Adaptation and Learning in Multi–Agent Systems', Vol. Lecture Notes in Artificial Intelligence Vol. 1042, Springer–Verlag.

Matarić, M. J. & Cliff, D. T. (1996), 'Challenges in Evolving Controllers for Physical Robots', *to appear in Robotics and Autonomous Systems.*

Moravec, H. P. & Cho, D. W. (1989), A Bayesian Method for Certainty Grids, *in* 'AAAI Spring Symposium on Robot Navigation', pp. 57–60.

Payton, D. (1990), Internalized Plans: a representation for action resources, *in* P. Maes, ed., 'Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back', MIT Press.

Payton, D., Keirsey, D., Kimble, D., Krozel, J. & Rosenblatt, K. (1992), 'Do Whatever Works: A robust approach to fault-tolerant autonomous control', *Journal of Applied Intelligence* **3**, 226–249.

Rosenschein, S. J. & Kaelbling, L. P. (1986), The Synthesis of Machines with Provable Epistemic Properties, *in* J. Halpern, ed., 'Theoretical Aspects of Reasoning About

Knowledge', Morgan Kaufmann, Los Altos, CA, pp. 83–98.

Schoppers, M. J. (1987), Universal Plans for Reactive Robots in Unpredictable Domains, *in* 'IJCAI-87', Menlo Park, pp. 1039–1046.

Simsarian, K. T. & Matarić, M. J. (1995), Learning to Cooperate Using Two Six–Legged Mobile Robots, *in* 'Proceedings, Third European Workshop of Learning Robots', Heraklion, Crete, Greece.

Steels, L. (1994), 'The Artificial Life Roots of Artificial Intelligence', *Artificial Life*.