Proceedings of CIRA'97, Monterey, CA, July 1997, pp. 124 - 129

Prior Structure for On-line Learning *

Manfred Huber and Roderic A. Grupen
Laboratory for Perceptual Robotics†
Department of Computer Science
University of Massachusetts, Amherst MA 01003

Abstract

On-line learning and adaptation capabilities are important for robot systems which operate in the real world and have to react to changes in the environment and the task requirements. Such learning schemes, however, often suffer from problems of complexity, rendering them intractable for on-line learning in complex domains. To address these problems, the approach presented here introduces prior structure which drastically reduces the size of the state and action spaces. Deriving control from a set of predictable control laws, the system is modeled more abstractly as a discrete event dynamic system which represents all possible system behavior. Use of this framework also permits the automatic incorporation of knowledge in the form of constraints, providing a basis for the explicit representation of safety conditions and "maturational" effects which can be used to further enhance learning performance. To illustrate this structure and to show its applicability, the acquisition of a rotation gait on a four-legged walking platform is demonstrated.

1. Introduction

Learning and adaptation are important for robot systems operating in uncertain environments. Such systems have to adjust to new task requirements and environmental conditions on-line without encountering catastrophic failures. In addition, autonomy requires that the learning method used has to be able to perform this adaptation without the need for outside supervision. In principle, exploration-based learning techniques, such as dynamic programming based reinforcement learning, are well suited for such tasks since they can learn from their own experiences and a simple reinforcement signal rather than from user input. These approaches have therefore been used for a range of learning applications [1, 14, 6] on simple systems and in off-line experiments. Without additional structure or locality assumptions, however, the high complexity of these learning methods renders them impracticable for on-line learning in complex robotic systems. In addition, these systems do not provide a way of introducing knowledge into the system to guide exploration or to ensure safety of the mechanism throughout the learning process.

To address the complexity issue and to start learning with a more robust system, these learning approaches have been combined with behavior-based control methods [2]. Here either the problem is decomposed *a priori* and only subproblems are learned [16, 10], or previously designed behaviors are used as actions within the learning framework [15, 17], reducing the complexity of the action and state spaces. While this makes more complex learning tasks feasible, the often procedural character of the behaviors still restricts their applicability to a limited domain and makes it necessary to design new behaviors or composition mechanisms when the task contingencies change. In addition, most such behaviors do not allow general predictions about their outcome and can thus not help to guide the exploration and ensure safety conditions.

One way to address safety issues is by using a parametric controller which is inherently stable and safe as the basis for the learning task [20]. In such a system, the underlying controller ensures a baseline performance while the learning component tunes the parameters for the given task. The use of a single controller, on the other hand, increases the designer effort required and limits the scope of the system. Another way of addressing safety is to provide a set of basic reflexes which are able to avert dangerous situations. This provides the system with an inherently safe behavioral control strategy from the start and uses learning in order to improve this policy [7]. These approaches, however, have been limited mainly to improving navigation strategies since such a set of reflexes is hard to find in general.

In a similar way as with safety considerations, most reinforcement learning schemes do not provide a mechanism to support the explicit incorporation of knowledge about the task and the robot mechanism into the learning and explo-

* This work was supported in part by NSF IRI-9503687
† http://piglet.cs.umass.edu:4321

ration process. While there have been attempts to incorporate such information implicitly in the form of shaping procedures [9], or by means of teaching [3], most such systems require all knowledge to be designed into the behaviors or composition mechanisms *a priori*.

The approach presented in this paper addresses the problems of complexity and safety, and provides a means of introducing domain knowledge by means of a hybrid discrete-continuous control structure. This structure uses a discrete event dynamic system (DEDS) supervisor on top of a set of primitive control laws which can be bound dynamically to different subsets of the system resources to form closed-loop controllers. Using the stable and convergent character of these controllers, the state space that has to be considered can be reformulated as a set of predicates representing the task objectives of the individual controllers. Built on this abstract state space, the DEDS supervisor is used as an interface to the learning component by enumerating all control alternatives that can be explored throughout the learning process. Employing the large number of formal techniques available within the DEDS framework, a supervisor can be synthesized automatically which incorporates a given set of constraints, effectively limiting the exploration to policies which obey those constraints, and thus guiding the learning process. Since these constraints can be revoked or changed at any time without changes in the learning structure, this mechanism not only permits the incorporation of safety constraints but can also be used to impose time varying constraints to explicitly guide the system through a sequence of developmental steps, or to change the resources available.

In the following, Section 2 gives a general overview of the interface to the learning component. Sections 3 and 4 describe the function and construction of the individual parts. Section 5, finally, shows the learning result achieved using this control structure in a quadruped walking example.

2. Control and learning structure

The approach presented here implements a control structure which acts as an interface between the physical world and the learning component. This structure reduces the size of the state space and provides a means to guide the exploration and thus to affect learning performance explicitly. Figure 1 shows the overall control structure used here.

In this framework, closed-loop controllers are constructed by associating system resources with primitive control laws. Behavior is then derived as a sequence of concurrent activations of such controllers. Designing the primitive control laws in a largely task and platform independent manner permits a small number of such control elements to span a potentially large number of tasks. In addition, the stable and convergent character of the closed-loop controllers implies that they form stable attractors in the underlying physical space, transforming this continuous space into a set of discrete system equilibria. The state of the system can thus be characterized abstractly by means of the predicates representing the functional goals of the individual controllers. Interpreting controller activation and convergence as events, the possible behavior of the system can be described in this abstract space using the DEDS framework.

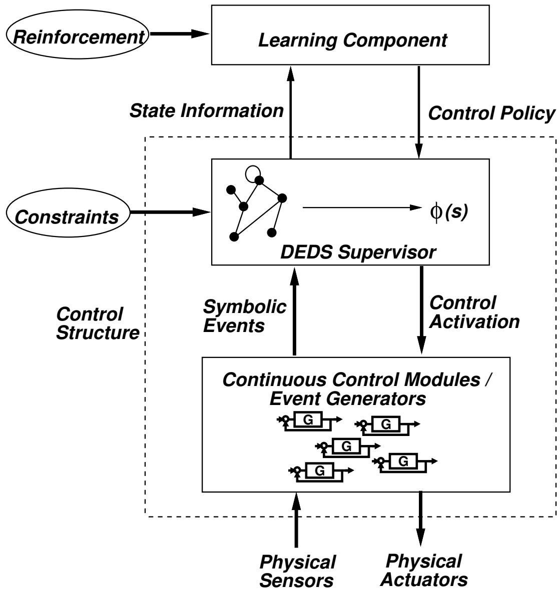


Figure 1. The control architecture

This DEDS model forms the basis for the learning component which acquires an optimal policy for the current task given the control alternatives offered by the DEDS supervisor. The use of this model reduces thus the size of the state space available to the policy formation process to the set of subgoals represented by the individual controllers. Moreover, the DEDS framework can be used to derive supervisors which incorporate given constraints automatically, effectively limiting exploration to policies which obey those constraints. Besides the possibility of shaping the learning system implicitly through changes in the task characteristics over time, this allows the imposition of safety and performance constraints, explicitly focusing learning on desired portions of the behavioral space. In addition, constraints can incorporate knowledge of kinematic limitations or be used to represent "maturational" effects by initially limiting the set of system resources or control alternatives considered. In the course of the learning process, such constraints can then be changed or removed whenever desired, permitting to guide the learning process.

The DEDS supervisor structure together with the controllers thus not only reduces the size of the state space but also guides the exploration and therefore directly influences the learning process.

3. The control basis

The control system presented in this paper interacts with the physical world by means of a set of closed-loop controllers. These controllers are derived on-line by associating input resources σ (sensors or sensor abstractions), and output resources τ (actuators) with primitive control laws Φ_i drawn from a control basis. Elements of this basis are carefully designed in order to yield stable and convergent controllers which allow for predictions about the outcome of controller activations and which form stable attractors within the underlying state space. In addition, since primitive control laws operate in generalized configuration spaces, they are largely task and device independent. This permits a small set of such control laws to span a wide variety of tasks.

Behavior is then constructed by the concurrent activation of closed-loop controllers $\Phi_i \frac{\sigma}{\tau}$. In order to preserve the stability and predictability properties of the individual controllers, concurrent activation is managed using the "subject to" ("$\triangleleft$") constraint. This constraint restricts the control actions of subordinate controllers such that they do not interfere with the objectives of higher priority controllers.

Different tasks in this framework are achieved by changing composition policies over the same set of controllers rather than by designing new control elements. In the walking example presented in this paper, the set of control laws consists of solutions to three generic robot control problems:

Φ_0: Configuration space motion control.
A harmonic function path controller [5] is used to generate robust, reactive, and collision-free motion through the configuration space of the robot.

Φ_1: Contact configuration control.
Contact configuration controllers [4] are employed to move foot contacts based on the local geometry of the environment in order to stabilize the robot.

Φ_2: Kinematic conditioning control.
The kinematic conditioning controller optimizes the posture of the articulated structure while it is engaged in an interaction with the world.

The resources available for the quadruped robot used in the experiments are the legs of the robot (legs 0, 1, 2, 3) and the location and orientation of the center of mass of the platform (x, y, φ). Closed-loop controllers can be derived by binding subsets of these system resources to a control law. Controller $\Phi_1 \frac{0,1,2}{0}$, for example attempts to establish a stable tripod stance on legs 0, 1, and 2 by moving leg 0.

This set of primitive control laws has already been used successfully for dextrous manipulation and locomotion tasks [8, 12]. In those cases, however, composition policies were hand crafted, requiring the designer to anticipate the exact behavior of each concurrent controller activation.

4. DEDS supervisor

In addition to the inherent reactivity of the controllers used in the control structure introduced here, their goal-directed and stable character enables them to partition the continuous state space into a discrete set of system equilibria which can be expressed by a set of functional predicates, each of which indicates whether the equilibrium state of the corresponding controller meets its control objective. In the walking example, for example, the predicate corresponding to the controller $\Phi_1 \frac{0,1,2}{0}$ indicates if the robot currently maintains a stable stance on legs 0, 1, and 2.

Predicates are only derived for elemental controllers and are independent of the output resources. Using the set of these predicates as the abstract state of the system, the possible effects of the activation of each elemental and composite controller on the state of the system can be determined [11]. The DEDS formalism can therefore be used to construct an abstract model of all possible system behavior in this abstract predicate space. As opposed to most DEDS approaches, it is not necessary for the designer to provide a complete system model [22, 13], rather this model can be constructed automatically from the abstract controller descriptions. The resulting DEDS supervisor, which takes the form of a nondeterministic finite state machine, forms the basic structure for the learning component. The abstraction from the continuous physical space to the abstract predicate space of the supervisor represents thereby a dramatic reduction in the complexity of the learning problem.

In addition to the state space reduction, the use of the DEDS framework also opens the derivation of such a supervisor to a large number of formal techniques [19, 21, 11]; In particular, it allows the imposition of constraints on the behavior of the system. The learning component operates on top of this supervisor which limits the exploration of the system to control policies which obey the given set of constraints, thus allowing the explicit incorporation of knowledge into the learning system. Furthermore, since these constraints can be altered or reversed throughout the learning process, they can be used to guide learning by changing the part of the behavior space which is accessible on-line.

4.1. Constraints for guiding learning

In most real world robotic tasks various kinds of information are available which can be used to guide the acquisition of robust control strategies and to improve learning performance. In order to incorporate such knowledge into the DEDS structure presented, it has to be formulated as constraints on the abstract system model.

A first kind of constraints that can be imposed limit the set of available resources and controllers. This allows the system to focus on certain subsets of the resources and to learn robust strategies for them before other parts of the physical structure can be used. This effectively limits the possible behavior of the system to relevant subsets of the functional state space, making it less complex to learn tasks which only involve these resources. Over time, these restrictions can then be removed, increasing the set of behavioral alternatives and permitting more tasks to be learned.

In the walking example presented here, such a resource constraint is incorporated by limiting the set of possible controllers and resource bindings to all instances of the contact configuration controller of the form $\Phi_1{}_{\underline{a}}^{a,b,c}$, where $a \neq b \neq c$ are legs of the robot, and one instance of the kinematic conditioning controller, $\Phi_2{}_{\underline{\varphi}}^{0,1,2,3}$. This effectively prevents any translational movement of the robot throughout the learning process. Using this set of 13 closed-loop controllers, the predicate space is limited to 5 predicates $(p_1, p_2, p_3, p_4, p_5)$ corresponding to the convergence of a controller-input binding pair in the following way:

$$p_1 \leftarrow \Phi_1{}_{\underline{*}}^{1,2,3}, \; p_2 \leftarrow \Phi_1{}_{\underline{*}}^{0,2,3}, \; p_3 \leftarrow \Phi_1{}_{\underline{*}}^{0,1,3},$$
$$p_4 \leftarrow \Phi_1{}_{\underline{*}}^{\overline{0},1,2}, \; p_5 \leftarrow \Phi_2{}_{\underline{*}}^{\overline{0},1,2,3},$$

where $\underline{*}$ is a wildcard and indicates the independence of the predicate evaluation from the output resource. Predicates $p_1 - p_4$ represent therefore stable three-legged stances on the corresponding input legs while p_5 evaluates the kinematic conditioning of the platform.

A second kind of constraints that can be incorporated consists of safety and performance constraints where the former characterize conditions that have to be met in order to avoid catastrophic failures, while the latter permit the incorporation of additional knowledge about characteristics of the given task. In the case of the walking example, a safety constraint for quasistatic walking can be imposed. This condition states that the platform has always to be in a stable configuration or in this example that at least one tripod stance has to be stable. In terms of the abstract predicate space, this constraint can then be formulated as $p_1 \vee p_2 \vee p_3 \vee p_4$.

A third kind of knowledge used can be represented in the form of intrinsic kinematic and domain constraints. Such constraints represent knowledge about the limitations of the actual mechanism or the domain it is operating in. For the four-legged walking platform employed, kinematic workspace limitations do not allow the simultaneous stability of two opposing support triangles which can be expressed in terms of the abstract predicates as $\neg(p_1 \wedge p_3) \wedge \neg(p_2 \wedge p_4)$.

Using these constraints and limiting the maximum number of concurrent controllers to three, the "$\triangleleft$" composition can be used to derive composite controllers, resulting in a set of 157 possible control alternatives. Furthermore, the DEDS supervisor shown in Figure 2 can be synthesized for the constrained system, where the numbers in the states represent the values of the predicate state vector $(p_0 p_1 p_2 p_3 p_4)$.

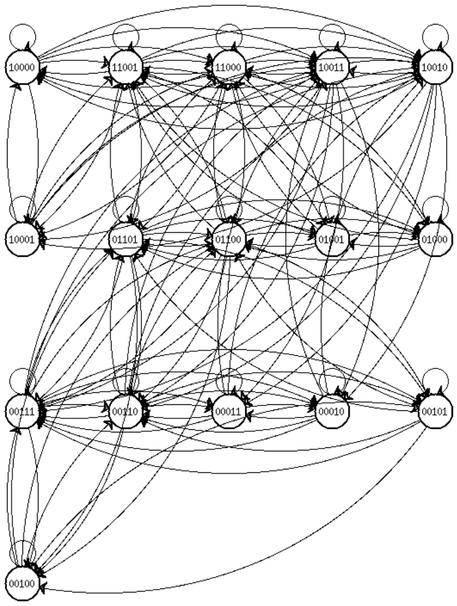


Figure 2. DEDS supervisor

This nondeterministic finite state supervisor has 16 states with an average of approximately 50 control alternatives available in each state, each of which can nondeterministically result in an average of approximately six successor states. This constrained DEDS supervisor forms the basis for the learning component which can explore the set of control alternatives represented in the model.

5. Learning component

Using the DEDS supervisor as the underlying structure for the learning problem, many different machine learning techniques can be applied, depending on the performance requirements. In the case of on-line learning in autonomous systems, exploration-based learning techniques such as reinforcement learning provide a good paradigm since they can learn from experience using a simple reward function.

In the control scheme presented here, the task of the learning component is to determine an optimal policy by selecting the best action with respect to the task requirements for each state. In addition, a second learning scheme can be used to acquire transition probabilities within the abstract DEDS model to improve the predictive power of the supervisor. This empirical model could then be used to perform off-line learning in order to reduce the amount of operation time required to learn a good policy for the given task [23, 18].

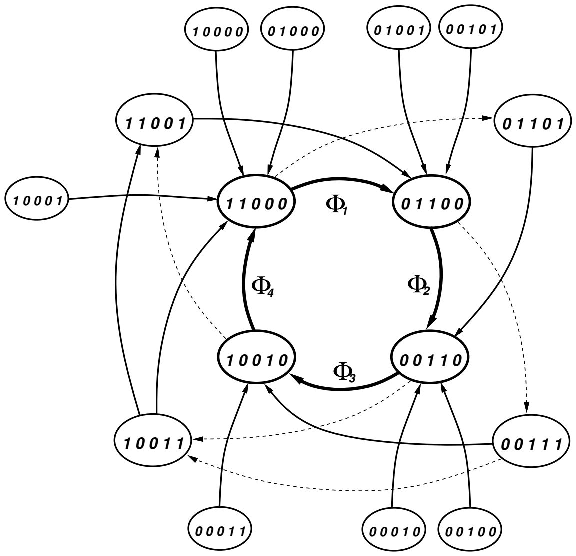
In the walking example used throughout this paper, Q-learning [24], a widely used reinforcement learning algorithm based on the temporal difference method is employed. This technique learns a value function over state-action pairs in order to estimate the discounted future payoff and thus the quality of a given action. The optimal control action in a given state is then simply the action with the highest Q-value. Throughout the exploration performed in order to determine this optimal policy, frequency counts are used to estimate the transition probabilities but no off-line learning is actually performed.

5.1. Learning results

To show the applicability and utility of the control structure described in this paper, it was applied to a four-legged walking robot. The example learning task involved the acquisition of a counterclockwise rotation gait on flat terrain using the control structure introduced in Sections 3 and 4, and a reinforcement signal proportional to the angular progress of the robot. Performing Q-learning on-line while exploration is slowly decreased from 100% to 10% random actions, the system rapidly acquired the correct gait pattern. On the physical robot and without off-line learning the turning gait was learned repeatably within approximately 12 minutes and without encountering an unsafe situation due to the constraints imposed on the DEDS supervisor. The minimum level of exploration was maintained throughout the learning phase to introduce perturbations in the behavior of the robot and thus to allow the system to learn a more robust policy over the full state space.

Figure 3 shows the policy learned for this task in the form of a nondeterministic finite state automaton. This graph shows all possible transitions that can occur while following the learned policy. The core cycle of this policy, indicated by bold transition arrows, corresponds to a stable turning gait. Transition probabilities within this cycle are higher than 95%, making it a stable attractor for this policy. For this core, the corresponding control actions are indicated on the bottom of the figure. The learned control actions in all other states attempt to lead the system onto this stable cycle, making the policy more robust with respect to perturbations.

Figure 4 shows the robot executing this central cycle of the learned policy which leads the system through a sequence of stable, three-legged stances. In each transition, a set of concurrent controllers is activated resulting in the simultaneous execution of body rotation and leg placement. Furthermore, the middle schematic, where circles correspond to foot locations and the cross indicates the location of the center of mass, shows that throughout execution the system always remains within the safe region of the state space.



$$\Phi_1 = \phi_{1\,\underline{1}}^{\underline{1,2,3}} \lhd \phi_{2\,\underline{\varphi}}^{\underline{0,1,2,3}} \lhd \phi_{1\,\underline{1}}^{\underline{0,1,3}}$$
$$\Phi_2 = \phi_{1\,\underline{2}}^{\underline{1,2,3}} \lhd \phi_{2\,\underline{\varphi}}^{\underline{0,1,2,3}} \lhd \phi_{1\,\underline{2}}^{\underline{0,1,2}}$$
$$\Phi_3 = \phi_{2\,\underline{\varphi}}^{\underline{0,1,2,3}} \lhd \phi_{1\,\underline{3}}^{\underline{1,2,3}}$$
$$\Phi_4 = \phi_{2\,\underline{\varphi}}^{\underline{0,1,2,3}} \lhd \phi_{1\,\underline{0}}^{\underline{0,2,3}}$$

Figure 3. Learned control policy for counterclockwise rotation task.

6. Summary and conclusions

For an autonomous robot system operating in an uncertain environment, learning and adaptation are important in order to adapt to changing environmental conditions and task requirements. Most such learning techniques, however, are expressed in high dimensional and continuous state spaces, rendering them intractable for on-line learning on complex systems. In addition these schemes do often not support the incorporation of outside knowledge to guide the learning process and to avoid unsafe situations.

The control system presented in this paper addresses these problems by means of the goal directed character of the underlying control laws and a DEDS model of the possible behavior of the system in terms of the functional predicates of the controllers. Use of the DEDS framework permits constraints on the behavior of the system to be imposed and provides a basis for restricting the system to safe control policies, as well as for expressing domain expertise. In addition, constraints can be changed or removed throughout the learning process, allowing a temporal sequence of constraints to be used to guide the exploration of the learning component over time. This together with the state space reduction achieved makes this approach suitable for on-line learning even in complex domains.

Initial experiments performed on a four-legged walking robot demonstrate that high performance on-line learning on complex platforms is possible with the control structure presented in this paper.

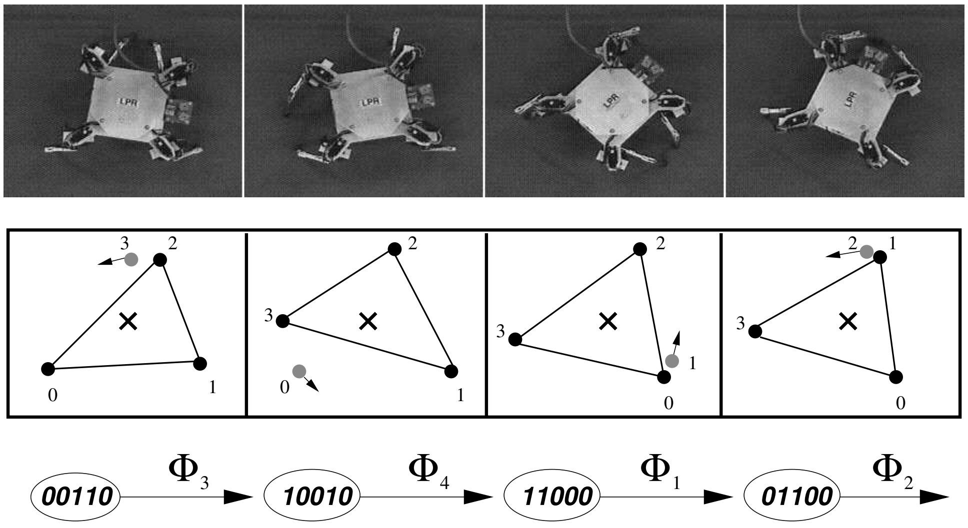


Figure 4. The robot executing the central gait cycle of the learned policy (top), the associated stability regions (middle), and the corresponding predicate state transitions (bottom).

References

[1] A. G. Barto, R. S. Sutton, and C. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man Cyber.*, 13(5):834–846, 1983.

[2] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE J. Robotics Automat.*, 2(1):14–23, Mar. 1986.

[3] J. A. Clouse. Learning from an automated training agent. In *Twelfth Int. Conf. Machine Learning, Workshop on Agents that learn from Other Agents*, Tahoe City, CA, 1995.

[4] J. A. Coelho Jr.. and R. A. Grupen. Effective multifingered grasp synthesis. Technical Report 94-12, CMPSCI Dept., Univ. of Mass., Amherst, Feb. 1994.

[5] C. I. Connolly and R. A. Grupen. The applications of harmonic functions to robotics. *J. Robotic Sys.*, 10(7):931–946, Oct. 1993.

[6] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*. Morgan Kaufmann, 1995.

[7] J. del R. Millán. Rapid, safe, and incremental learning of navigation strategies. *IEEE Trans. Syst. Man Cyber.*, 26(3):408–420, 1996.

[8] R. A. Grupen, M. Huber, J. A. Coelho Jr., and K. Souccar. Distributed control representation for manipulation tasks. *IEEE Expert*, 10(2):9–14, Apr. 1995.

[9] V. Gullapalli. *Reinforcement Learning and its Application to Control*. PhD thesis, Univ. of Mass., Amherst, MA, 1982.

[10] J. Hoff and G. Bekey. An architecture for behavior coordination learning. In *Proc. 1995 IEEE Int. Conf. Neural Networks*, pages 2375–2380, Perth, Australia, Nov. 1996.

[11] M. Huber and R. A. Grupen. A hybrid discrete event dynamic systems approach to robot control. Technical Report 96-43, CMPSCI Dept., Univ. of Mass., Amherst, Oct. 1996.

[12] M. Huber, W. S. MacDonald, and R. A. Grupen. A control basis for multilegged walking. In *Proc. Int. Conf. Robotics Automat.*, pages 2988–2993, Minneapolis, MN, Apr. 1996.

[13] J. Košecká and L. Bogoni. Application of discrete event systems for modeling and controlling robotic agents. In *Proc. Int. Conf. Robotics Automat.*, pages 2557–2562, San Diego, CA, May 1994.

[14] L.-J. Lin. *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, Carnegie Mellon Univ., Pittsburgh, PA, Jan. 1993.

[15] P. Maes and R. Brooks. Learning to coordinate behaviors. In *Proc. 1990 AAAI Conf. on Artificial Intelligence*.

[16] S. Mahadevan and J. Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, 55:311–365, 1992.

[17] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1).

[18] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13, 1993.

[19] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proc. IEEE*, 77(1):81–97, Jan. 1989.

[20] S. Singh, C. Connolly, R. Grupen, and A. Barto. Robust reinforcement learning in motion planning. In *Advances in Neural Information Processing Systems 6 (NIPS)*, 1994.

[21] M. Sobh, J. Owen, K. Valvanis, and D. Gracani. A subject-indexed bibliography of discrete event dynamic systems. *IEEE Robotics & Automation Magazine*, 1(2):14–20, 1994.

[22] J. Stiver, P. Antsaklis, and M. Lemmon. A logical approach to the design of hybrid systems. *Mathematical and Computer Modelling*, 27(11/12):55–76, 1996.

[23] R. S. Sutton. First results with Dyna, an integrated architecture for learning, planning and reacting. In W. T. Miller III, R. S. Sutton, and P. J. Werbos, editors, *Neural Networks for Control*, pages 179–189. MIT Press, 1990.

[24] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge Univ., Cambridge, England, 1989.