15^{th} **Int. Joint Conf. on Artificial Intelligence, Nagoya, Japan, August 1997, pp. 1366 - 1371**

Learning to Coordinate Controllers - Reinforcement Learning on a Control Basis*

Manfred Huber and Roderic A. Grupen
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

Autonomous robot systems operating in an uncertain environment have to be reactive and adaptive in order to cope with changing environment conditions and task requirements. To achieve this, the hybrid control architecture presented in this paper uses reinforcement learning on top of a Discrete Event Dynamic System (DEDS) framework to learn to supervise a set of basis controllers in order to achieve a given task. The use of an abstract system model in the automatically derived supervisor reduces the complexity of the learning problem. In addition, safety constraints may be imposed a priori, such that the system learns on-line in a single trial without the need for an outside teacher. To demonstrate the applicability of the approach, the architecture is used to learn a turning gait on a four legged robot platform.

1 Introduction

Autonomous robot systems operating in an uncertain environment have to be able to cope with new situations and task requirements. Important properties of the control architecture of such systems are thus that it is reactive, allows for flexible responses to novel situations, and that it adapts to longer lasting changes in the environment or the task requirements.

Although model-based control techniques have been used successfully in a wide variety of tasks, they are very sensitive to imprecisions in the model and are often not robust with respect to unexpected situations. To better address the reactivity requirements of autonomous systems, behavior-based architectures [Brooks, 1986] were developed. In this paradigm, system behavior is constructed on-line from combinations of elemental, reactive behaviors. The often ad hoc character of these behaviors, however, can lead to an extremely complex organization of behavioral elements. In addition, the resulting policy can be brittle with respect to relatively minor perturbations including those introduced by other behaviors or by changes in control context. The control basis approach used here attempts to circumvent this problem by employing carefully designed declarative control primitives to construct overall system behavior, and therefore allows predictions about the outcome of behavioral sequences. This approach has been used successfully in manipulation and locomotion tasks [Grupen *et al.*, 1995; Huber *et al.*, 1996].

While such bottom-up approaches address the issue of reactivity, the used composition is in most cases given by the designer and very specific to the task at hand. In order to render such a system adaptive and allow it to adjust its overall behavior to changing task requirements, learning techniques have to be employed. In the extreme case, this learning has to occur without the direct influence of an outside teacher in order to obtain autonomous behavior. Reinforcement learning techniques are well suited to such behavior composition tasks since they can learn sequences of behavior from simple reinforcement signals. In most applications, however, these techniques have been applied at a very low level, thus leading to a very high complexity of the learning task. This complexity rendered these approaches inadequate for on-line learning in complex systems.

To address these complexity problems as well as to provide a base reactivity to the system, some work has been done to combine this learning framework with the robustness of behavior based control approaches [Maes and Brooks, 1990; Mahadevan and Connell, 1992]. Here either the problem is decomposed a priori and only subproblems are learned, or previously designed behaviors are used as elemental actions within a reinforcement learning task. While this dramatically reduces the complexity of the state and action spaces for the learning problem, the character of the behaviors often restricts their applicability to a very limited domain, potentially

*This work was supported in part by NSF IRI-9503687

requiring the design of new components whenever the task changes. In addition, most of these approaches do not address safety considerations by allowing the exploration to take random actions, thus permitting the occurrence of catastrophic failures. In autonomous systems, however, this is not permissible since the system can not recover. It is thus necessary that such systems can learn in a single trial without the need for outside supervision. One way to address this is by using a parametric controller which is inherently safe as a basis for the learning task [Singh *et al.*, 1994]. The learning component learns thereby only a setting of the parameters while the controller assures a baseline performance. The use of a single controller, however, increases designer effort and limits the scope of the system.

The approach presented here addresses the complexity and safety issues by means of a hybrid continuous/discrete control architecture. Behavior is constructed on-line from a set of stable and convergent control elements. The stable character of these base controllers is then used in a Discrete Event Dynamic System (DEDS) framework [Sobh *et al.*, 1994] to construct a supervisor which provides the structure for the reinforcement learning component. This dramatically reduces the complexity of the learning problem by reducing the state and action spaces, and supports techniques designed to limit exploration to safe and relevant areas of the behavior space. Moreover, the system inherits the reactivity and stability of the underlying control basis.

In the following, Section 2 briefly introduces the control basis approach before Section 3 describes the discrete event architecture used to automatically synthesize admissible control policies, and the reinforcement learning technique for acquiring a policy for a given task. Section 4, finally, shows an example for the overall architecture in the walking domain where a turning gait is learned on-line in a single trial.

2 The Control Basis Approach

The control basis approach constructs behavior online by combining feedback control elements drawn from a set of carefully designed basis controllers. Individual control elements are thereby largely task and device independent and represent solutions to generic robot control problems, allowing a small set of these elements to span a large range of tasks on a wide variety of platforms.

In this framework, control is derived on-line by associating input resources σ (sensors or sensor abstractions), and output resources τ (actuators) with feedback control laws Φ_i drawn from the control basis. The resulting controllers $\Phi_i \frac{\sigma}{\tau}$ can then be activated concurrently according to a task dependent composition policy under the "subject to" ("$\triangleleft$") constraint. This constraint restricts the control actions of subordinate controllers such that they do not counteract the objectives of higher priority controllers. The resulting concurrent control policy inherits the stability and convergence properties of the elemental control elements. A complete control policy takes then the form of a sequence of concurrent controller activations of the form shown in Figure 1. Different tasks in this framework are achieved by changing composition policies over the same set of controllers rather than by designing new control elements.

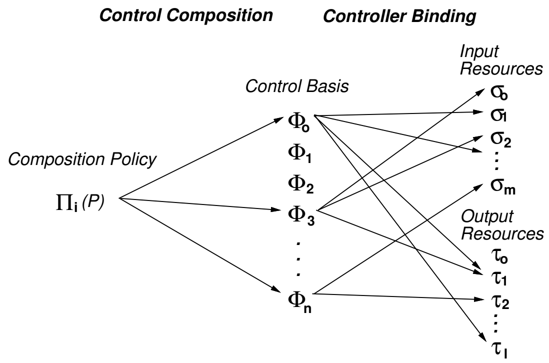


Figure 1: Control Composition

This approach has already been used successfully on a variety of tasks in the manipulation and locomotion domain [Grupen *et al.*, 1995; Huber *et al.*, 1996]. In all these cases, however, composition policies were hand crafted, thus requiring the system designer to anticipate the exact behavior of the controllers. To achieve more autonomous behavior of a robot system, however, the system has to be able to adapt to novel situations and task contingencies without the need for outside supervision. In order to achieve this, the architecture presented in this paper learns the optimal composition policy in an efficient way using the reinforcement learning paradigm.

3 Composition Architecture

Reinforcement learning [Barto *et al.*, 1993] offers a flexible way to acquire control strategies automatically and thus to adapt to new contingencies and task requirements. Most reinforcement learning systems, however, operate at a very low level by directly influencing actuator commands, easily leading to an explosion in the complexity of the learning task. This renders such approaches impractical for on-line learning in complex systems. To avoid this problem and to be able to prevent catastrophic failures, the architecture presented here attempts to learn a composition policy for the underlying controllers. To do so it uses the goal directed character of the control modules described in Section 2 to build an abstract description of the system under the influence of the basis controllers. Running controllers to convergence, the possible behavior of the system is modeled as a DEDS on a symbolic predicate space characterized by the individual goals of the control modules. This abstract system model is then used as the basis for an exploration-based learning system to acquire an optimal control strategy for the given task. The DEDS formalism encodes safety constraints in the model and thus limits the exploration to the space of admissible control policies. The overall architecture is shown in Figure 2.

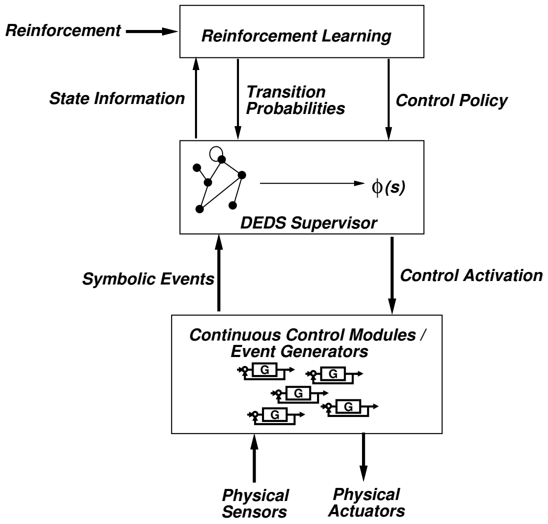


Figure 2: The Control Architecture

In this approach, all continuous sensor input and actuator output is handled by the elements of the control basis. Activation and convergence of these controllers is then interpreted as the events used in the abstract DEDS model of the system behavior. After imposing safety and domain constraints, the DEDS supervisor represents the set of all admissible control policies. Throughout exploration or in a separate system identification phase, additional transition probabilities for this model can be acquired, thus improving the quality of the largely task independent model. This model allows knowledge of system behavior to be generalized across tasks and supports additional off-line learning on the estimated system model [Sutton, 1990]. In addition to acquiring intrinsic structure in the form of transition probabilities, this hybrid architecture also solves the resource allocation problem by learning to assign resources to controllers optimally given the current reinforcement structure.

3.1 DEDS Supervisor

The feedback control primitives employed as elemental actions in this approach act as stable attractors and thus form basins of attraction in the continuous physical space. The system may therefore be described at an abstract level by means of convergence predicates associated with the underlying controllers. The abstraction from continuous state space to discrete predicate space represents a dramatic reduction in complexity and thus forms a good basis for the reinforcement learning task. Since activation and convergence determine the progress of the plant in this abstract space, the overall system can be modeled as a hybrid DEDS, opening the derivation of a possible supervisor to a large body of formal techniques [Ramadge and Wonham, 1989; Sobh *et al.*, 1994]. In particular, constraints such as safety and the absence of deadlock conditions can be imposed a priori on the control policy.

Predicate Space Description

In order to allow for the abstraction step and to provide an efficient way of deriving an abstract model of the possible system behavior, the effects and interactions between control primitives have to be described symbolically by their possible effects on the set of predicates. Characterizations for composite controllers can thereby be generated automatically from descriptions of the individual elements of the control basis, thus limiting the work of the designer to these control elements. As opposed to most DEDS approaches where the designer has to provide a complete system model [Stiver *et al.*, 1996; Košecká and Bogoni, 1994], these simple descriptions allow for an automatic generation of a predicate space model of all possible system behavior (For details on the controller characterization and the construction of the system model see [Huber and Grupen, 1996]). This model takes the form of a nondeterministic finite state automaton, and forms the basis for the supervisor synthesis and reinforcement learning components of the proposed architecture.

Supervisor Synthesis

In the DEDS framework, control is performed by enabling and disabling of controllable events in a supervisor. In the case of the architecture presented here, the set of these events corresponds to controller activations while convergence events are not controllable. To allow for safety of a chosen control policy or to ensure that no deadlock occurs, the DEDS formalism provides methods to automatically impose constraints on the supervisory automaton. Doing so, the system model can be pruned a priori to the set of all policies which obey the given set of constraints [Ramadge and Wonham, 1989; Huber and Grupen, 1996]. In the same fashion, additional domain knowledge and designer preferences can be incorporated into the control system, further reducing the complexity for the reinforcement learning component. Throughout learning and system operation, this discrete supervisor is then used to limit exploration to admissible parts of the behavior space, and to activate controllers according to the policy selected by the learning system.

3.2 Learning Component

Reinforcement learning provides a mechanism for adapting to changing environments and tasks without external supervision [Mahadevan and Connell, 1992]. In this exploration-based paradigm, the system learns a control policy which maximizes the amount of reinforcement it receives. A major drawback of this scheme, however, is its inherent complexity and the need for exploration in order to find better policies. In the architecture presented here, these problems are addressed by learning controller activations on top of the predicate space model defined by the DEDS supervisor, thus reducing the size of the action and state spaces considered in the learning task, and enforcing safety constraints throughout exploration. In addition, this also facilitates the acquisition of transition probabilities between predicate states and thus allows run-time experience to improve the predictive power of the abstract model.

The learning component used here employs Q-learning [Watkins, 1989], a widely used temporal difference method that learns a value function over state/action pairs in order to represent the quality of a given action. Estimates of the future discounted payoff of an action,

$$Q(x,a) = E(r + \gamma \max_{b \in A} Q(y,b))$$

are computed, where r is the immediate reinforcement obtained at this time step, and γ is a discount factor. Using this function the control policy is given as the action with the highest Q-value in the current state. To adapt this to the finite state transition model used as the underlying state space description for learning, this value function can be represented in a distributed fashion as

$$Q(x,a) = \sum_{y \in X} (p(x,a,y)\mathcal{Q}(x,a,y)),$$

where $p(x,a,y)$ is the probability that controller a in state x will lead to state y and $\mathcal{Q}(x,a,y)$, represents the value of the corresponding transition. Throughout learning this estimate is then updated according to

$$Q_t(x,a) = (1-\beta)Q_{t-1}(x,a) + \beta(r_t + \gamma \max_{b \in A} Q_{t-1}(y,b))$$

where β is the learning rate. At the same time, frequency counts can be used to determine the transition probabilities within the nondeterministic supervisor. Together this allows the overall architecture to adapt efficiently to changing task requirements by on-line acquiring correct control policies.

4 Locomotion Example

To demonstrate the applicability of the proposed architecture, it has been implemented on a four legged, twelve degree of freedom walking robot. The objective was to acquire useful policies for turning gaits. It has already been shown that hand crafted control gaits for such tasks can be derived [Huber *et al.*, 1996]. The example presented here uses the proposed architecture to learn solutions autonomously with minimal external supervision. To do this, the robot was put in an initial stable configuration onto an even surface and the learning process was initiated without any further input from an outside teacher.

4.1 Control Basis and DEDS Supervisor

The control basis used for these tasks was already used successfully for dextrous manipulation and locomotion tasks [Grupen *et al.*, 1995; Huber *et al.*, 1996] and consists of solutions to three generic robotics problems,

Φ_0: Configuration space motion control,

Φ_1: Contact configuration control, and

Φ_2: Kinematic conditioning.

Each of these basis control laws can then be bound to subsets of the system resources (legs $0,1,2,3$ and position and orientation of the center of mass x, y, φ) shown in Figure 3. For details on this control basis and the resource bindings see [Huber *et al.*, 1996].

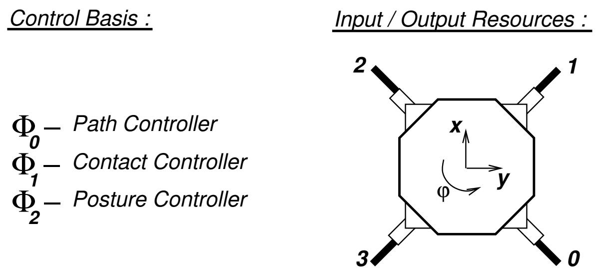


Figure 3: Controller and Resource Notation

For the example presented here, the set of possible controllers and controller bindings was limited in order to allow for a concise notation for the predicate space model. The set of controller/resource combinations available to the system consists here of all instances of the contact configuration controller of the form $\Phi_1 \frac{a,b,c}{\underline{a}}$, where $a,b,c \in \{0,1,2,3\}$, $a \neq b \neq c \neq a$ are three legs of the robot, and one instance of the kinematic conditioning controller, $\Phi_2 \frac{0,1,2,3}{\underline{\varphi}}$. Using this set of 13 elemental candidate controllers, the $\lhd$ constraint can be used to construct composite controllers, allowing a total of 157 composite controllers or actions in the DEDS and learning components. In addition, this choice of candidate controllers limits the predicate space 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:

$$p_1 \leftarrow \Phi_{1\,\underline{*}}^{\;\underline{1,2,3}} \quad , \; p_2 \leftarrow \Phi_{1\,\underline{*}}^{\;\underline{0,2,3}} \quad , \; p_3 \leftarrow \Phi_{1\,\underline{*}}^{\;\underline{0,1,3}} \; ,$$
$$p_4 \leftarrow \Phi_{1\,\underline{*}}^{\;\underline{0,1,2}} \quad , \; p_5 \leftarrow \Phi_{2\,\underline{*}}^{\;\underline{0,1,2,3}} \quad ,$$

where $\underline{*}$ is a wildcard and indicates the independence of the predicate evaluation from the output resource.

After automatically constructing the graph of all possible system behavior in this space, a safety constraint for quasistatic walking, namely that the platform has to be always stable, can be imposed on the supervisor. In terms of the predicates this implies that at least one stance has to be stable, or in other words $p_1 \vee p_2 \vee p_3 \vee p_4$ has to evaluate true. Furthermore, knowledge about the platform can be introduced in the form of domain constraints. For the legged platform employed here, for example, kinematic limitations do not allow the simultaneous stability of two opposing support triangles. Adding this knowledge as $\neg(p_1 \wedge p_3) \wedge \neg(p_2 \wedge p_4)$, further reduces the size of the supervisor to the one shown in Figure 4 ,

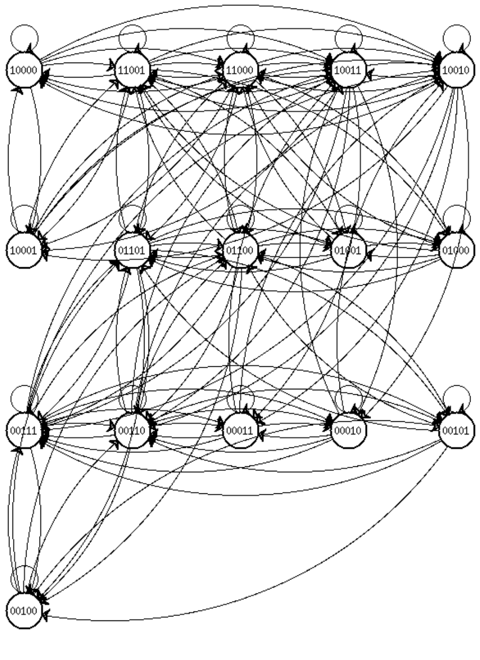


Figure 4: DEDS Supervisor for Rotation Task

where the numbers in the states represent the values of the 5 predicates. It should be noted here, that for the purpose of illustration, the complete supervisor has been built a priori in this example. In general, however, this could be done incrementally in the course of exploration without the violation of any constraints.

4.2 Learning Results

The supervisor derived from the control basis represents all admissible control policies. It does not, however, express any task objectives or an optimal control policy. In order to learn these for the counterclockwise rotation task, a reinforcement schedule has to be present which rewards the system whenever it performs the task correctly. The reinforcement used in this example is 1 if the control action led to a counterclockwise rotation, -1 if it led to a clockwise rotation, and 0 otherwise. The robot system is then put onto a flat surface and the learning process is started.

Several experiments of this form were performed in order to investigate the performance of the control and learning components. In all these trials the system rapidly acquired the correct gait pattern while exploration was slowly decreased from 100% to 10%. This minimum level of random actions was maintained to introduce perturbations and thus to learn a more robust policy. Figure 5 depicts a typical learning profile for this task.

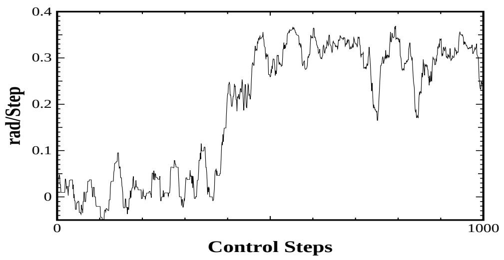
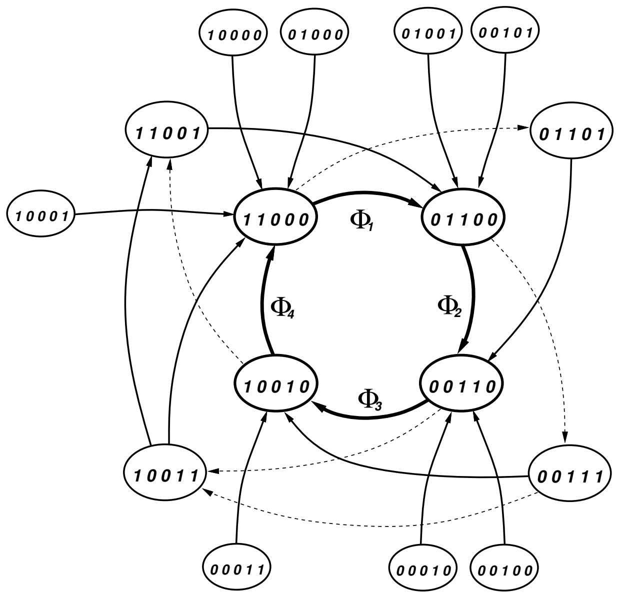


Figure 5: Learning Curve for Counterclockwise Task

The learning curve shows that a correct turning gait is learned after approximately 500 learning steps. Throughout this entire learning process, which on the real robot took approximately 15 minutes, the robot platform never entered an unsafe situations due to the limitations imposed by the DEDS supervisor.

The final control policy is shown in Figure 6. This graph shows all possible transitions that can occur while following the learned policy. The core of this policy is the cycle indicated by bold transition arrows which corresponds to a stable turning gait. Transition probabilities within this cycle are $> 95\%$, 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.



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

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

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

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

Figure 6: Learned Control Policy for Counterclockwise Rotation Task

5 Conclusions

Reactive and adaptive control architectures for autonomous systems pose many challenges due to the complexity of the task and the limited amount of supervision possible. The architecture presented in this paper employs a hybrid control architecture with a reinforcement learning component in order to address these issues. Continuous reactive control is thereby derived from a carefully designed control basis while the composition policy is learned on top of an abstract predicate space in a DEDS framework. This allows the imposition of safety constraints and thus permits new tasks to be learned online in a single trial and without the need for an external teacher. In addition, it dramatically reduces the complexity of the action and state space, making learning feasible even for complex tasks and platforms.

References

[Barto *et al.*, 1993] A.G. Barto, S.J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. Technical Report 93-02, CMPSCI Dept., Univ. of Mass., Amherst, MA, 1993.

[Brooks, 1986] R.A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

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

[Huber and Grupen, 1996] 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, MA, Oct. 1996.

[Huber *et al.*, 1996] M. Huber, W.S. MacDonald, and R.A. Grupen. A control basis for multilegged walking. In *Proc. Int. Conf. Robotics Automat.*, pages Vol.4 2988–2993, Minneapolis, MN, April 1996.

[Košecká and Bogoni, 1994] 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.

[Maes and Brooks, 1990] P. Maes and R. Brooks. Learning to coordinate behaviors. In *Proceedings of the 1990 AAAI Conference on Artificial Intelligence*. 1990.

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

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

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

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

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

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

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