NIPS'98 Workshop : Abstraction and Hierarchy in Reinforcement Learning, Breckenridge, CO. ©1998 LPR

Learning Robot Control - Using Control Policies as Abstract Actions[*]

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

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. In the extreme case, this learning has to occur without the direct influence of an outside teacher, making the reinforcement learning paradigm an attractive option since it allows to learn sequences of behavior from simple reinforcement signals [1, 17]. However, while these techniques have been applied to simple robot systems and in simulation[2, 5, 7, 10, 11, 12, 6], the complexity of the primitive action and state spaces of most robots leads to a need for large amounts of experiences to learn a given task, thus rendering these methods impracticable for on-line learning on such systems. Furthermore, most such learning systems do not provide a means for introducing a priori knowledge, thus permitting the occurrence of catastrophic failures which is often not permissible in real world systems which have to learn new tasks in a single trial. To address these issues, the control architecture presented here uses more abstract actions which allow to define the system as a Discrete Event Dynamic System (DEDS) on an abstract, discrete state space, within which a policy for the given task is learned. To illustrate this, the architecture has been applied to walking tasks on a four-legged walking robot.

The use of abstract actions within the reinforcement learning framework[14] promises to make it possible to address more complex tasks and platforms. Much of this promise stems from the possibility to treat the resulting system as an event driven system rather than a clock driven one, reducing the set of points at which the learning agent has to consider a new action to the times when certain control or sensor events happen. While this allows for optimal decision points to be missed if the corresponding sensor signals lie outside the scope of the current set of control and sensor alternatives, it also leads to a focus of attention and can dramatically

[*] This work was supported by NSF IRI-9503687, IRI-9704530, and CDA-9703217

reduce the amount of exploratory experience required to learn a good policy. Another strength of abstract actions is their potential to bridge hidden state, allowing certain aspects of the state to be ignored, and thus permitting policies to be learned in a smaller, more abstract state space.

The abstract actions used in the approach presented here are stable and convergent closed-loop control policies. This implies that they divide the underlying physical space into a set of stable regions within which they drive the robot system towards an attractor. This attractor, in turn, can be characterized abstractly by means of predicates indicating the achievement of the functional goals of the associated abstract actions. If abstract actions are executed until convergence, the behavior of the system can largely be described by these attractors, which therefore allow to transform the underlying continuous space into a set of discrete system equilibria. Using the convergence of abstract actions as control events, the behavior of the system can thus be modeled approximately as a hybrid DEDS with a discrete state space corresponding to the convergence predicates of the abstract actions. While this state abstraction might produce hidden state, its action dependent choice should ensures that the discrete space encompasses all tasks directly addressable by the underlying abstract actions. This DEDS then forms the basic substrate for the reinforcement learning problem and, through the formal techniques available in the DEDS framework[15, 8] and local models of the behavior of the individual abstract actions, allows constraints to be imposed a priori in order to limit exploration to safe and relevant control alternatives. Control alternatives available to the DEDS and learning systems are thereby the abstract actions, as well as the hierarchical, concurrent activation of multiple of these abstract actions using the "subject to" ("$\triangleleft$") constraint. This constraint prioritizes the control actions such that a lower priority action can not counteract the progress of a higher priority one and thus ensures that the stability and convergence properties of the original abstract actions are inherited by the composite actions. Using this, the learning component learns a control policy which optimizes the given reinforcement, as well as an improved abstract system model in terms of the transition probabilities within the DEDS model. This overall architecture is shown in Figure 1.

As shown in this figure, all direct sensory input and actuator output in this approach is handled by the abstract actions in the bottom layer. Activation and convergence of these individual or composite actions are then interpreted as discrete events in the abstract DEDS model of possible system behavior which forms the basis for the reinforcement learning system. A priori constraints imposed on this model can be used to limit the range of possible actions to keep the system within a safe mode of operation, as well as to implement temporally varying "maturational" constraints to improve learning performance. In addition to this, this structure also promises the possibility of hierarchical action spaces since learned control policies, together with the corresponding predicate space models, could be included as abstract actions into the learning process. While this leads to an increase in the size of the potential action space and thus implies that methods have to be found which effectively select actions that are relevant for the task at hand, it could also dramatically increase the efficiency of the learning system for more complex tasks and would also result in the introduction of more abstract predicates and state descriptions.

To illustrate this learning and control architecture, the following shows an example of the overall architecture applied to a four-legged walking robot, where a turning

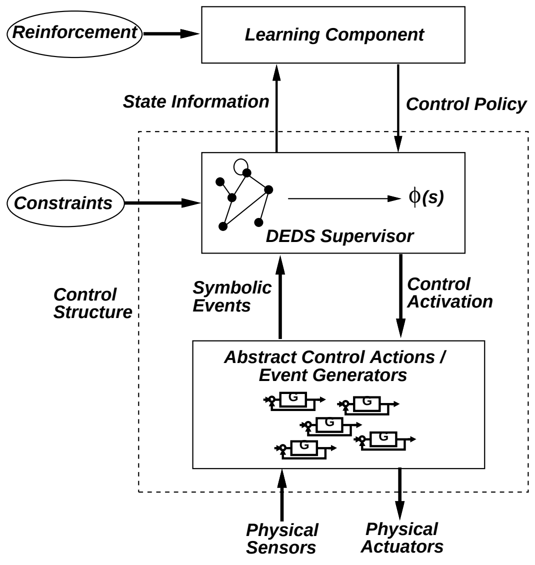


Figure 1: The Control Architecture

gait is learned on-line in a single trial. Locomotion gaits are formed here as sequences of concurrent activations of a set of feedback controllers and represented as nondeterministic finite state machines. The set of feedback controllers, which represents the abstract actions used at the bottom layer of the architecture, is here formed using a control basis approach. In this approach controllers are established by attaching a set of input resources (sensor abstractions) and output resources (abstract actuators) to a control law which addresses a generic control objective. In the case of the locomotion tasks, three control laws are used:

Φ_0**: Configuration space motion control -** a harmonic function path controller is used to generate collision-free motion of the robot in configuration space[4].

Φ_1**: Contact configuration control -** contact controllers locally optimize the stability of the foot pattern based on the local terrain[3].

Φ_2**: Kinematic conditioning control -** a kinematic conditioning controller locally optimizes the posture of the legs.

Each of these control laws Φ_i can be bound on-line to input resources σ and output resources τ derived as subsets of the system resources (legs $0, 1, 2, 3$ and position and orientation x, y, φ) of the four-legged robot illustrated in Figures 2 and 3.

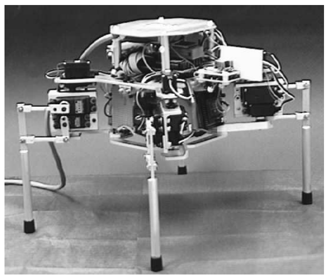


Figure 2: Walking Robot

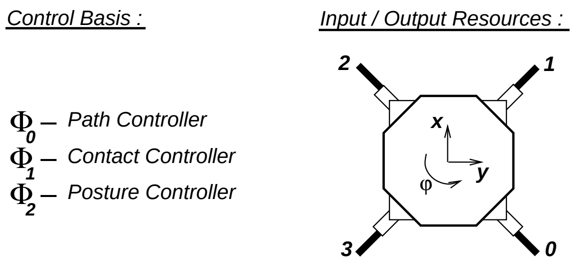


Figure 3: Controller and Resource Notation

The resulting feedback controllers $\Phi_i\frac{\sigma}{\underline{\tau}}$ can be activated concurrently under the "subject to" ("$\triangleleft$") constraint. The composite controller $\Phi_2\frac{0,1,2,3}{\underline{\varphi}} \triangleleft \Phi_1\frac{0,1,2}{\underline{0}}$, for example, attempts to achieve a stable stance on legs 0, 1, and 2 by moving leg 0 with the dominant controller while the subordinate controller optimizes the kinematic posture of all four legs within the "nullspace" of Φ_1 by rotating the body. For the example presented here, the set of possible controllers was limited in order to allow for a concise notation for the predicate space model. The set of abstract actions 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 abstract actions, the "$\triangleleft$" constraint can be used to construct a total of 157 actions available to the DEDS and learning components. In addition, this choice of abstract actions limits the set of convergence predicates to 5 elements $(p_1, p_2, p_3, p_4, p_5)$ since multiple abstract actions have identical control objectives and their predicates can thus be combined. The 5 predicates correspond to the convergence of abstract actions in the following way:

$$p_1 \leftarrow \Phi_1\frac{1,2,3}{\underline{*}}\ ,\ p_2 \leftarrow \Phi_1\frac{0,2,3}{\underline{*}}\ ,\ p_3 \leftarrow \Phi_1\frac{0,1,3}{\underline{*}}\ ,\ p_4 \leftarrow \Phi_1\frac{0,1,2}{\underline{*}}\ ,\ p_5 \leftarrow \Phi_2\frac{0,1,2,3}{\underline{*}}\ ,$$

where $\underline{*}$ is a wildcard and indicates the independence of the predicate evaluation from the output resource. These predicates, together with initial, abstract models of the behavior of the abstract actions, form then the basis of the DEDS system which represents the space of all possible system behavior. The DEDS framework allows then to impose a quasistatic walking constraint of the form $p_1 \vee p_2 \vee p_3 \vee p_4$ (at least one stance has to be stable at all times) to determine the set of admissible actions in each of the abstract predicate states.

To address new tasks, Q-learning[17] is used here to acquire a control policy for a given reinforcement signal on top of the constrained DEDS model. This scheme allows the acquisition of control policies even if their objective is not represented as a state in the underlying state space, and thus permits cyclic policies. In the experiment presented here an immediate reinforcement proportional to the rotational progress, $r_t = \varphi_t - \varphi_{t-1}$, is used to acquire a counterclockwise rotation gait. The safety constraint imposed in the DEDS layer allows thereby to simply start the robot in an arbitrary configuration on a flat surface and to learn the policy on-line in a single trial. A characteristic learning curve for this task is shown in Figure 4.

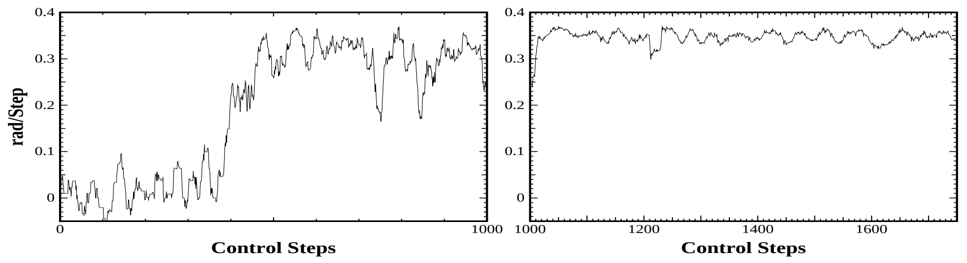
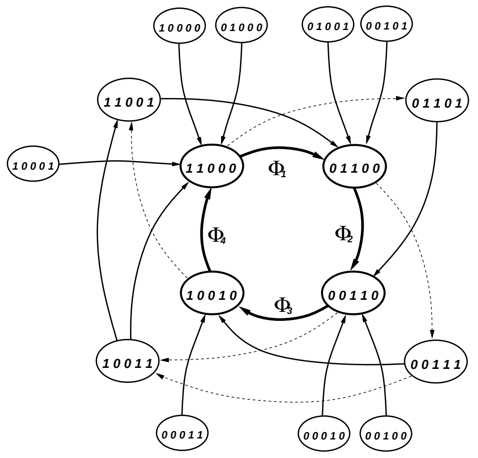


Figure 4: Learning Curve for Counterclockwise Rotation Task (left) and Performance of the Learned Policy without Exploration (right)

This graph, in which a control step indicates one controller activation, i.e. one transition in the DEDS model, shows that the robot rapidly acquires a good policy. The complete learning task executes on the real platform in approximately 11 minutes.

At the same time that such a policy is learned, the exploration can also be used to estimate transition probabilities between predicate states and thus to improve the abstract model of the system behavior. Such a model can be useful for off-line learning[16, 13], as well as to allow the transfer of the learned control policies into the space of abstract actions. Figure 5 shows the learned policy and the corresponding system model.



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

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

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

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

Figure 5: Learned Rotation Gait

Here the numbers in the states represent the values of the 5 predicates, the controller definitions on the right indicate the learned policy for the core of the turning gait, and the width of the transitions indicates the acquired transition probabilities, with bold arrows for the central gait cycle indicating probabilities greater than 98%. The execution of this central cycle on the real robot is also depicted in Figure 6.

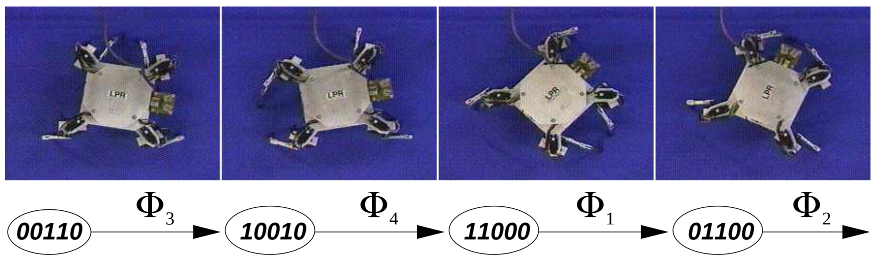


Figure 6: The Robot Executing the Central Gait Cycle of the Learned Policy (top) and the Corresponding Predicate State Transitions (bottom)

This and other locomotion experiments performed using this control architecture[9] show that the use of abstract actions together with a DEDS layer which allows to incorporate certain types of a priori knowledge into the system and permits action dependent state abstraction, represents a feasible approach to perform reinforcement learning for more complex tasks on-line on real robots.

References

[1] A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. Technical Report 93-02, University of Massachusetts, Amherst, MA, 1993.

[2] 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.

[3] J. A. Coelho Jr. and R. A. Grupen. A control basis for learning multifingered grasps. *J. Robotic Sys.*, 14(7):545–557, October 1997.

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

[5] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In *NIPS 8*. Morgan Kaufmann, 1995.

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

[7] V. Gullapalli. *Reinforcement Learning and its Application to Control.* PhD thesis, University of Massachusetts, Amherst, MA, 1982.

[8] 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, October 1996.

[9] M. Huber and R. A. Grupen. A control structure for learning locomotion gaits. In *Seventh International Symposium on Robotic and Applications*, Anchorage, AK, May 1998. TSI Press.

[10] L.-J. Lin. *Reinforcement Learning for Robots Using Neural Networks.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, January 1993.

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

[12] M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

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

[14] D. Precup and R. S. Sutton. Multi-time models for temporally abstract planning. In *NIPS 10*, pages 1050–1056, Denver, CO, December 1997. MIT Press.

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

[16] 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.

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