AN ALGEBRAIC APPROACH TO ABSTRACTION IN REINFORCEMENT LEARNING

A Dissertation Presented

by

BALARAMAN RAVINDRAN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2004

Computer Science

AN ALGEBRAIC APPROACH TO ABSTRACTION IN REINFORCEMENT LEARNING

A Dissertation Presented

by

BALARAMAN RAVINDRAN

To Prof. S. Banumoorthy

ACKNOWLEDGMENTS

I am very grateful to my thesis advisor Andy Barto. His guidance, support and patience were instrumental in bringing this work to fruition. He taught me how to be a better researcher and how to better express my ideas. He keeps an open mind while approaching a research question and has constantly surprised me by the connections he can draw with existing literature. When I started working on my thesis, I would return from my meetings with Andy with a different book each week on topics such as potential fields, category theory, and statistics, from all of which Andy drew relevant ideas. I will always strive to emulate him as a researcher. Thank you Andy for everything.

The other members of my committee also provided me with great support and encouragement. I thank Rod Grupen for the feedback on drafts of my thesis, and for helping me better understand symmetry groups and the control basis approach; Neil Berthier for his support, discussions on developmental psychology, and advice on certain ethical questions; and Sridhar Mahadevan for many stimulating discussions on hierarchy and abstractions. My interaction with Rich Sutton and Robbie Moll during the first two years here were very educative and exposed me to two very different ways of approaching research problems. Their enthusiasm when confronted with an interesting problem is infectious.

Many others have contributed to the development of the ideas in the thesis and to my understanding of the issues involved. I thank Bob Givan and Matthew Greig for clarifying certain points about their minimization framework. I thank Rob Platt for numerous discussions on applying my ideas to practical problems and for commenting on several drafts of the thesis. I thank Mike Rosenstein for exposing me to a wide variety of work in RL and related fields. I also thank Ted Perkins, Mohammad Ghavamzadeh, Khashayar Rohanimanesh, and Dan Bernstein for many hours of interactions, which were either stimulating, entertaining, outrageous, vacuous or all of them simultaneously.

The erstwhile Adaptive Networks Lab, currently the Autonomous Learning Lab, is an excellent and enjoyable environment for research. I thank all the denizens of the lab, past and present, for making it so. I especially thank Amy McGovern and Andy Fagg for being such good friends and for their constant support and encouragement. I also thank the lab members for sitting through numerous practice talks and for giving useful feedback. Again, special thanks to Andy Barto, Andy Fagg, Amy and Mike for attending more than a fair share of these mind-numbing, repetitive sessions. My presentation skills have improved largely due to their feedback. In addition I thank Nathan Sitkoff, Doina Precup, Leo Zelevinsky, Anders Jonsson, Mance Harmon, Matthew Schlesinger, Marc Pickett, Jad Davis, Ash Shah, Özgür Şimşek, Pippin Wolfe, Randy Casstevens, Sarah Osentoski, Vicky Manfredi, and Andrew Stout, who over the years have helped maintain a great atmosphere in the lab.

My stay in the graduate school in UMass has been made that much more enjoyable by the excellent support staff, who really care about the students. Thank you for all the help. I especially thank Gwyn Mitchell and Pauline Hollister, for responding readily to my numerous requests for help. I thank Sharon Mallory, the graduate program manager, for her understanding and support. When I applied for admission to UMass, my application package was delivered to a different school by the courier. It was due to the efforts of Sharon and Rosalyn Black, at MIT, that my application even reached here. I thank Rod Grupen and Neil Immerman the past and present graduate program directors, and Robbie Moll, for their understanding of my unique circumstance and enabling me to meet my goals.

During my initial years in this country, I received a lot of support from my cousin Ravi G. Chandran. Thank you Ravi, for all the wonderful conversations on life, the universe and everything. My parents have been very supportive of me in all my endeavors, even if they did not want me to be this far away from them. Their love and their confidence in my abilities have always provided me with the necessary impetus to succeed. I also thank my parents-in-law for their support and encouragement. Vibhu—enthralling, exciting, doughty, exasperating, wonderful, ethereal—the world is no longer the same after you came, and neither are my priorities. Thank you for reminding me to smell the flowers. Without my wife Suba, none of this would have been possible. Even during the darkest of times, she was steadfast in her belief and provided me with the motivation to persevere. Thank you for the patience and all the love.

This material in this work is based upon work supported by the National Science Foundation under Grant No. ECS-9980062 to Andrew G. Barto and Grant No. ECS-0218125 to Andrew G. Barto and Sridhar Mahadevan. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

ABSTRACT

AN ALGEBRAIC APPROACH TO ABSTRACTION IN REINFORCEMENT LEARNING

FEBRUARY 2004

BALARAMAN RAVINDRAN

B.E., MADURAI-KAMARAJ UNIVERSITY, INDIA

M.Sc.(Engg.), INDIAN INSTITUTE OF SCIENCE, BANGALORE, INDIA

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Andrew G. Barto

To operate effectively in complex environments learning agents require the ability to form useful abstractions, that is, the ability to selectively ignore irrelevant details. Stated in general terms this is a very difficult problem. Much of the work in this field is specialized to specific modeling paradigms or classes of problems. In this thesis we introduce an abstraction framework for Markov decision processes (MDPs) based on homomorphisms relating MDPs. We build on classical finite-state automata literature and develop a minimization framework for MDPs that can exploit structure and symmetries to derive smaller equivalent models of the problem. Since employing homomorphisms for minimization requires that the resulting abstractions be exact, we introduce approximate and partial homomorphisms and develop bounds for the loss that results from employing relaxed abstraction criteria.

Our MDP minimization results can be readily employed by reinforcement learning (RL) methods for forming abstractions. We extend our abstraction approach to hierarchical RL, specifically using the options framework. We introduce relativized options, a generalization of Markov sub-goal options, that allow us to define options without an absolute frame of reference. We introduce an extension to the options framework, based on relativized options, that allows us to learn simultaneously at multiple levels of the hierarchy and also employ hierarchy-specific abstractions. We provide certain theoretical guarantees regarding the performance of hierarchical systems that employ approximate abstraction. We empirically demonstrate the utility of relativized options in several test-beds.

Relativized options can also be interpreted as behavioral schemas. We demonstrate that such schemas can be profitably employed in a hierarchical RL setting. We also develop algorithms that learn the appropriate parameter binding to a given schema. We empirically demonstrate the validity and utility of these algorithms. Relativized options allow us to model certain aspects of deictic or indexical representations. We develop a modification of our parameter binding algorithm suited to hierarchical RL architectures that employ deictic representations.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS .... **v**

ABSTRACT .... **viii**

LIST OF TABLES .... **xiii**

LIST OF FIGURES .... **xiv**

CHAPTER

1. INTRODUCTION **1**

1.1 Motivation 1
1.2 Outline of Thesis 4

2. BACKGROUND AND NOTATION **9**

2.1 Markov Decision Processes 9
2.2 Semi-Markov Decision Processes 11
2.3 Reinforcement Learning 11
2.4 Partitions, maps and equivalence relations 12

3. MDP HOMOMORPHISMS AND MINIMIZATION **15**

3.1 Group Homomorphism 15
3.2 Machine Homomorphism 17
3.3 MDP Homomorphism 18
3.4 Minimization Framework 20
3.5 Identifying Homomorphisms 26
3.6 Relation to Stochastic Bisimulations 31
3.7 Learning With Reduced Models 34
3.8 Related Work 36

4. SYMMETRY, STRUCTURE AND APPROXIMATIONS 40

4.1 Modeling Symmetries 40

4.1.1 Symmetry Groups 41
4.1.2 Discussion on Identifying Symmetry Groups 46
4.1.3 Related Work 47

4.2 Homomorphisms of Factored MDPs 49

4.2.1 Structured MDPs 50
4.2.2 Structured Morphisms 52
4.2.3 Permutation Symmetry Groups 54
4.2.4 Discussion 57

4.3 Approximate Equivalence 58

4.3.1 Approximate Homomorphisms 59
4.3.2 Bounding the Approximation Loss 61
4.3.3 Bounded Approximate Homomorphisms 64
4.3.4 Discussion 67

5. ABSTRACTION IN HIERARCHICAL SYSTEMS 69

5.1 SMDP Homomorphisms 70
5.2 Partial Homomorphisms 72
5.3 Sub-goal Options 74
5.4 Relativized Options 78
5.5 Hierarchical Problem Decomposition 80

5.5.1 Hierarchical Policies 82
5.5.2 Learning with Hierarchical Decompositions 85

5.6 Illustrative Example 87
5.7 Approximate Equivalence 90
5.8 Relation to MAXQ Safe State-abstraction 93
5.9 Related Work 94
5.10 Summary 97

6. OPTION SCHEMAS AND DEIXIS 98

6.1 Relativized Options as Option Schemas 98
6.2 Related Work 100
6.3 Choosing Transformations 102

6.3.1 A Bayesian Algorithm 104

6.3.2 Experimental Illustration 105

6.4 Handling Approximate Equivalence 107
6.5 Experiments in a Complex Game Environment 109
6.6 Deictic Representation 118

6.6.1 Modeling Aspects of Deixis with Relativized Options 120
6.6.2 Experimental Illustration in a Complex Game Environment 123
6.6.3 Perceptual Aliasing with Deictic Representations 128
6.6.4 Related Work 130
6.6.5 Discussion 132

6.7 Applications of Homomorphisms in the Control Basis Framework 133

6.7.1 Control Basis Framework 134
6.7.2 Designing Option Schemas 136
6.7.3 Discussion 141

7. CONCLUSIONS 143

7.1 Summary 143
7.2 Future Work 146
7.3 Closing Remarks 149

APPENDICES

A. PROOF OF THEOREM 5 150
B. SOME RESULTS ON APPROXIMATIONS OF DYNAMIC PROGRAMS 152

BIBLIOGRAPHY 156

LIST OF TABLES

Table **Page**

3.1 Transition probabilities for the MDP $\mathcal{M}$ shown in Figure 3.4(a): (i) under action a_1. (ii) under action a_2. 25

3.2 The transition probabilities of the MDP $\mathcal{M}'$ shown in Figure 3.4(b). 25

4.1 Algorithm for finding simple projection homomorphisms assuming that the MDP is completely specified. 54

4.2 Incremental algorithm for constructing the $\mathcal{G}$-reduced image given MDP $\mathcal{M}$ and some $\mathcal{G} \leq \mathrm{Aut}\mathcal{M}$. Q is the queue of states to be examined. This algorithm terminates when at least one representative from each equivalence class of $\mathcal{G}$ has been examined. 57

6.1 Parameter settings in various rooms in Figure 6.6. 112

6.2 Subset of controllers for accomplishing two handed grasps of large objects. 138

LIST OF FIGURES

Figure **Page**

1.1 (a) A blocks world task. The blocks are described by their color, position and label. (b) One abstract representation that ignores the labels. (c) Another abstract representation that ignores the colors. 2

1.2 (a) A symmetric gridworld problem. The goal state is G and there are four deterministic actions. State-action pairs (A, E) and (B, N) are equivalent in the sense described in the text. (b) A reduced model of the gridworld in (a). The state-action pairs (A, E) and (B, N) in the original problem both correspond to the pair $(\{A, B\}, E)$ in the reduced problem. A solution to this reduced gridworld can be used to derive a solution to the full problem. 5

3.1 A Group Homomorphism represented by Commutative Diagrams 16

3.2 An FSA homomorphism represented by commutative diagrams. 18

3.3 An MDP Homomorphism represented by Commutative Diagrams 19

3.4 (a) Transition graph of example MDP $\mathcal{M}$. This MDP is irreducible under a traditional minimization framework. Our notion of homomorphic equivalence allows us to minimize this further. (b) Transition graph of the minimal image of the MDP $\mathcal{M}$ in (a). 24

3.5 (a) Transition graph of example MDP $\mathcal{M}$. Repeated from Figure 3.4(a). (b) Transition graph of the quotient MDP $\mathcal{M}|B$. See text for description. Note that this is *isomorphic* to the MDP in Figure 3.4(b). 30

4.1 (a) A symmetric gridworld problem. Reproduced from Figure 1.2. (b) Reflection of the gridworld in (a) about the *NE-SW* diagonal. 42

4.2 (a) Transition graph of a symmetric MDP. (b) Transition graph of a similar MDP, but with a trivial symmetry group. (c) Minimal image for both the MDPs. 45

4.3 (a) The DBN of the coffee robot domain described in the text. The *a* node is the action node and can take values *dc, go, gu, nop.* (b) The DBN of the homomorphic image generated by a simple projection on to a subset of features. 52

4.4 Towers of Hanoi. The task is to move the disks from the darker position to the lighter position. Panels (a), (b) and (c) show situations that are equivalent under permutation symmetries. 55

4.5 (a) A slightly asymmetric gridworld problem. The goal state is G and there are four deterministic actions. The problem is approximately symmetric about the NE-SW diagonal. (b) A reduced model of the gridworld in (a). The state-action pairs (A, E) and (B, N) in the original problem both correspond to the pair $(\{A, B\}, E)$ in the reduced problem. A solution to this reduced gridworld can be used to derive an approximate solution to the full problem. 59

4.6 (a) A spatial navigation problem. The goal is to reach the shaded region in the center. The environment is approximately symmetric about the dotted lines. (b) An approximate homomorphic image of the task in (a). Transitions into the lightly shaded regions are determined either by aggregate dynamics or specified as intervals. See text for more details. 61

4.7 (a) Optimistic choice of parameters for the BMDP shown in Figure 4.6(b). (b) Pessimistic choice of parameters for the same BMDP. In both the figures, dark squares are obstacles and the goal is to reach the shaded area in the lower right. See text for explanation of parameter choices. 66

5.1 (a) A gridworld task with *rooms* and the usual gridworld dynamics. The dark square indicates the goal. The lighter squares are alternate locations for the goal. (b) A homomorphic image when the dark square is the goal. The goal in this image is the dark triangle at the bottom. The transitions wrap around the dotted lines, i.e., actions W and S at the left edge will cause transitions to the right edge and action E and N at the right edge cause transitions to the left edge. (c) A partial homomorphic image restricted to the *room states*. The dark oval is an absorbing state. 73

5.2 (a) A simple rooms domain with similar rooms and usual stochastic gridworld dynamics. The task is to collect all 5 objects (black diamonds) in the environment and reach the central corridor. The shaded squares are obstacles. (b) The option MDP corresponding to a *get-object-and-leave-room* option. See text for full description. 77

5.3 The modified option MDP corresponding to the *get-object-and-leave-room* relativized option. The lightly shaded squares in the middle of the room indicate states with a negative reward. The dashed line indicates the optimal policy for this modified option MDP. The dotted line indicates a component of the hierarchically optimal policy, when using this option in the task described by Figure 5.2 83

5.4 (a) Comparison of asymptotic performance of various learning agents on the task shown in Figure 5.2. See text for description of the agents. (b) Comparison of initial performance of the regular and relativized agents on the same task. 90

5.5 Comparison of the rate of improvement to final performance of the regular and relativized agents on the task shown in Figure 5.2. 91

5.6 (a) Comparison of asymptotic performance of the regular and relativized agents on the modified rooms task. See text for description of the task. (b) Comparison of initial performance of the two agents on the same task. 92

5.7 (a) A simple rooms domain with dissimilar rooms. The task is to collect all 5 objects in the environment. (b) The option BMDP corresponding to a *get-object-and-leave-room* option. See text for full description. 93

5.8 (a) Comparison of asymptotic performance of the regular and relativized agents on the task in Figure 5.7. (b) Comparison of initial performance of the two agents on the same task. 94

6.1 Comparison of initial performance of agents with and without knowledge of the appropriate partial homomorphisms on the task shown in Figure 5.2 with various levels of stochasticity. 107

6.2 Typical evolution of posteriors for a subset of transforms in Room 5 in Figure 5.2, with a slip of 0.1. 108

6.3 A simple rooms domain with dissimilar rooms. The task is to collect all 5 objects in the environment. 110

6.4 Comparison of initial performance of agents with and without knowledge of the appropriate partial homomorphisms on the task shown in Figure 5.7 with a slip of 0.1. 111

6.5 Typical evolution of weights for a subset of transforms in Room 5 in Figure 5.7, with a slip of 0.1. 112

6.6 A game domain with interacting adversaries and stochastic actions. The task is to collect all 4 objects, the black diamonds, in the environment. The adversaries are of two types—benign (shaded) and delayers (black). See text for more explanation. 113

6.7 The option MDP corresponding to the sub-task *get-object-and-leave-room* for the domain in Figure 6.6. There is just one delayer in the option MDP. The state is described by the x and y coordinates of the agent and the delayer and a boolean feature indicating possession of the diamond. 115

6.8 Comparison of the performance of an agent with 4 regular options and an agent using a relativized option and no knowledge of the correct transformation on the task shown in Figure 6.6(a). The option MDP employed by the relativized agent is shown in Figure 6.6(b). 117

6.9 Typical evolution of weights for a subset of transformations in Room 4 in Figure 6.6(a), with a slip of 0.1. 118

6.10 Typical evolution of weights for a subset of transformations in Room 2 in Figure 6.6, with a slip of 0.1. 119

6.11 Deixis in a simple blocks world domain: The task in (a) is to place block B on A, while in (b) is to place X on Z. The two tasks reduce to that of placing block pointed to by $\times$ over block pointed to by $+$. 121

6.12 A modified game domain with interacting adversaries and stochastic actions. The task is to collect the black diamond. The adversaries are of three types—benign (shaded), retriever (white) and delayers (black). See text for more explanation. 124

6.13 The option MDP corresponding to the sub-task *get-object-and-leave-room* for the domain in Figure 6.12. There is just one delayer and one retriever in this image MDP. 125

6.14 Typical evolution of a subset of weights of the monolithic agent on the task shown in Figure 6.12. 127

6.15 (a) Typical evolution of a subset of the *delayer* weights of the deictic agent on the task shown in Figure 6.12. (b) Typical evolution of a subset of the *retriever* weights on the same task. 128

6.16 Perceptual aliasing in a simple blocks world domain: Both (a) and (b) map to the same representation and are treated as equivalent. But the task is to stack blocks of three different colors and the desired action in the two situations are different. 129

6.17 The UMass Torso holding a ball in a two handed grasp. 134

6.18 The UMass Torso holding a ball by opposing gravity with the left hand. 137

6.19 The transition structure of a family of sub-tasks for manipulating objects in the workspace. A state labeled ${}^X\Phi_j$ means that the controller ${}^X\Phi_j^j$ has converged in that state. 139

6.20 A relativized option that represents the family of tasks shown in Figure 6.19. A state labeled ${}^X\Phi$ means that for some j, the controller ${}^X\Phi_j^j$ has converged in that state. 140

6.21 A relativized option that represents the *regrasp before reach* sub-task. A state labeled ${}^X\Phi_i$ means that the controller ${}^X\Phi_i^i$ has converged in that state, where i depends on the transformation chosen. 142

CHAPTER 1
INTRODUCTION

1.1 Motivation

The ability to form abstractions is one of the features that allow humans to operate effectively in complex environments. We systematically ignore information that we do not need for performing the immediate task at hand. While driving, for example, we ignore details regarding clothing and the state of our hair. On the other hand, while preparing to attend a ball, we would want to pay special attention to our clothing and hair. Researchers in artificial intelligence (AI), in particular machine learning (ML), have long recognized that applying computational approaches to operating in complex and real-world domains requires that we incorporate the ability to handle and form various abstractions.

Researchers in many fields, ranging from various branches of mathematics to social network analysis, also recognize the utility of abstractions and have tried to answer questions such as what is a useful abstraction and how to model abstractions. Abstract representations keep recurring in various guises in the literature. For example, statisticians use the notion of *sufficient statistic*, which is a function of the observed data that summarizes the data so as to provides enough information for solving the problem at hand, such as determining the probability of occurrence of a certain event. Informally, one can define a good abstraction to be a function of the observable features of a task such that it is a "sufficient statistic".

Let us consider the blocks world task shown in Figure 1.1(a). There are a set of blocks described by their color, location and label. Suppose the goal is to obtain a

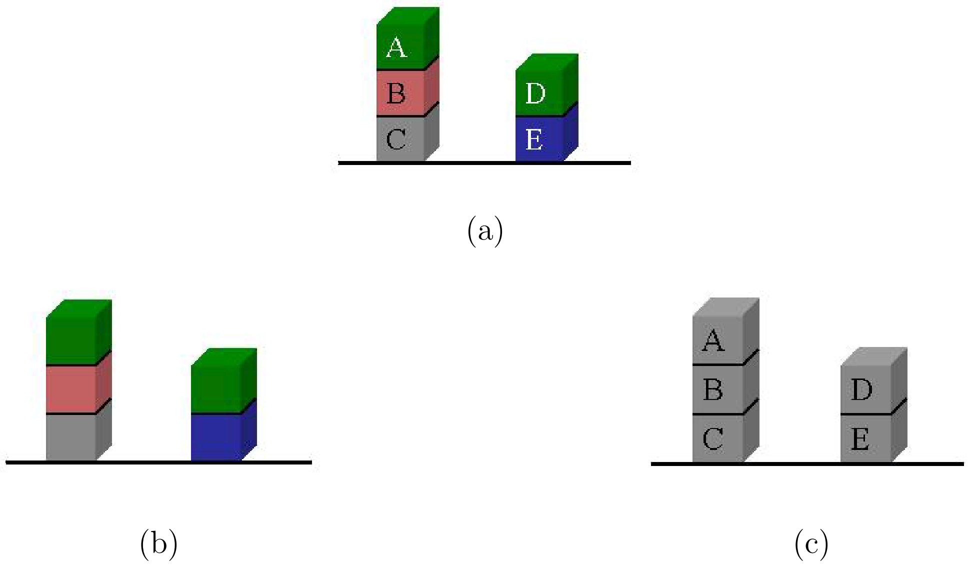


Figure 1.1. (a) A blocks world task. The blocks are described by their color, position and label. (b) One abstract representation that ignores the labels. (c) Another abstract representation that ignores the colors.

green colored block. Then the label of the block is irrelevant and a representation that pays attention only to the color and the position of the blocks is sufficient (Figure 1.1(b). On the other hand, if the goal is to obtain the block labeled B, the color of the block is irrelevant and a representation with only the label and the position is sufficient (Figure 1.1(c)). Thus the notion of sufficiency varies with the goal.

Determining sufficiency and providing ways of modeling abstractions are well studied problems in AI (e.g., Amarel, 1968; Popplestone and Grupen, 2000; Dean and Givan, 1997; Knoblock, 1990; Dean and Lin, 1995). They are also difficult problems when stated in general terms. Therefore, much of the work in this field is specialized to particular classes of problems or specific modeling paradigms. In this work we focus on Markov decision processes (MDPs) (Puterman, 1994), a formalism widely employed in modeling and solving stochastic sequential decision problems.

Our goal in this thesis is to develop a general framework that can accommodate different notions of abstractions employed with MDPs, including simple aggregation (Boutilier and Dearden, 1994; Sutton and Barto, 1998), symmetries (Zinkevich and Balch, 2001; Popplestone and Grupen, 2000), structured abstractions (Dean and Kanazawa, 1989; Boutilier et al., 1999, 1995, 2001), projections and other feature selection mechanisms. There is a large body of research in algebraic abstraction algorithms for other modeling paradigms in the literature (e.g., Hartmanis and Stearns, 1966; Kemeny and Snell, 1960; Lee and Yannakakis, 1992; Whitt, 1978). We build on this wealth of experience and develop a framework that provides additional intuition into existing MDP abstraction approaches and extends them in ways not envisioned earlier.

In particular, the algebraic approach we develop is amenable to modeling abstractions in reinforcement learning (RL) systems. Reinforcement learning (Sutton and Barto, 1998) refers to a collection of learning techniques for approximate solution of stochastic sequential decision problems and are often employed with MDP models of the problems. RL techniques offer many advantages over other approximate solution methods, such as maintaining a close relation to classical MDP solution methods, and the ability to learn in real-time and focus on parts of the problem that are most relevant. Not all RL algorithms require complete models of the environment and frequently employ some form of abstraction and/or function approximation to speed up learning, unlike many conventional approaches. They can also work with extensions to the MDP formalism such as Semi-Markov decision processes (SMDPs) and partially observable MDPs (POMDPs). Recent advances have led to hierarchical learning algorithms that significantly broaden the applicability of RL (Sutton et al., 1999; Parr and Russell, 1997; Dietterich, 2000a). Our abstraction framework extends to hierarchical settings in a natural way.

1.2 Outline of Thesis

Our approach to MDP abstraction is based on the notion of *MDP homomorphisms*. This is an extension of machine homomorphisms from the finite state automata (FSA) literature (Hartmanis and Stearns, 1966). Machine homomorphisms help establish precise correspondences between automata that have similar behavior and identify states that can be aggregated together to derive "smaller" equivalent models. We extend the notion to MDPs by incorporating decision making and stochasticity. But the power of our approach comes from employing *a state-dependent action recoding*. This enables us to apply our results to a wider class of problems and extend existing MDP abstraction frameworks in ways not possible earlier.

Our approach to abstraction belongs to the class of algorithms known as model minimization methods. The goal of model minimization is to derive a reduced model representation in which some key property of the original model is preserved. In the case of MDPs, we require that some aspects of the dynamic structure of the MDP is preserved in the reduced model. We show that this is sufficient to allow us to solve the original problem modeled by the MDP. The MDP minimization approach we present in this thesis can be viewed as an extension of the MDP minimization framework proposed by Dean and Givan (Dean and Givan, 1997; Givan et al., 2003). This earlier framework considers equivalence of states based on the notion of stochastic bisimulations (Larsen and Skou, 1991), whereas we believe that homomorphisms are a simpler notion than bisimulations and provide better insight into the minimization process. Our proposed framework also supports powerful extensions to the minimization process such as exploiting symmetries of MDPs.

To illustrate the concept of minimization, consider the simple gridworld shown in Figure 1.2(a). The goal state is labeled G. Taking action E in state A is equivalent to taking action N in state B, in the sense that they go to equivalent states that are both one step closer to the goal. One can say that the state-action pairs (A, E) and (B,

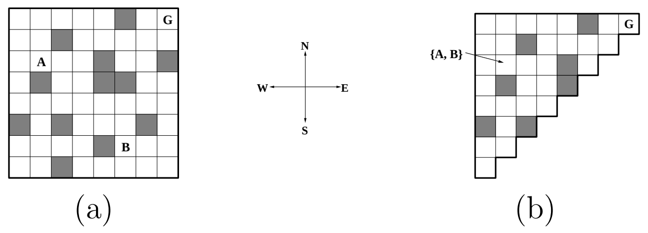


Figure 1.2. (a) A symmetric gridworld problem. The goal state is G and there are four deterministic actions. State-action pairs (A, E) and (B, N) are equivalent in the sense described in the text. (b) A reduced model of the gridworld in (a). The state-action pairs (A, E) and (B, N) in the original problem both correspond to the pair $(\{A, B\}, E)$ in the reduced problem. A solution to this reduced gridworld can be used to derive a solution to the full problem.

N) are equivalent. One can exploit this notion of equivalence to construct a smaller model of the gridworld (Figure 1.2(b)) that can be used to solve the original problem.

Figure 1.2 illustrates a situation in which the symmetry in the problem is exploited in the abstraction, yet existing MDP minimization approaches do not explicitly accommodate such symmetric reductions. Symmetries of a structure are characterized traditionally by the *symmetry group* of the structure. This is the group of mappings of the structure onto itself, such that some structural property is preserved. For example, in the gridworld in Figure 1.2(a), such a mapping is given by reflecting the states about the NE-SW diagonal and flipping actions N and E, and actions S and W. This leaves the transition structure of the gridworld unaltered. We incorporate this traditional group-theoretic definition into our framework to model symmetries of MDPs. The goal of minimization methods is to derive the minimal model (or the smallest model) equivalent to the given MDP. In general this is an NP-hard problem. Symmetry in the problem definition introduces additional structure that can be exploited to derive compactly represented reductions in systems with some inherent structure. For example, in Figure 1.2, the transformation can be expressed simply as an exchange of the x and y co-ordinates and of the suitable pair of actions. We exploit the additional structure associated with symmetries to derive a polynomial-time algorithm for minimization for structured problems modeled as *factored MDPs* (Dean and Kanazawa, 1989).

While abstractions that lead to exact equivalences are very useful, they are often difficult to achieve. In fact, to apply our approach to real-world problems we would need to consider a variety of "relaxed" minimization criteria. For example, in the grid-world in Figure 1.2 assume that the action E succeeds with probability 0.9 and the action N succeeds with probability 0.8. When actions fail, you stay in the same cell. We could still consider (A, E) and (B, N) equivalent for minimization purposes. We explore various relaxations of our minimization framework to accommodate *approximate equivalence* of state-action pairs. We use results from Whitt (1978) and Givan et al. (2000) to bound the loss in performance resulting from our approximations. We also address issues of learning with approximate reduced models.

In large complex problem domains, it is often difficult to identify reduced models of the entire problem. In such cases it is useful to consider *partial equivalences* that do not hold over all parts of the state-action space. For instance, while navigating in a building, there might be many rooms that can be treated equivalently, while each wing in the building is unique and has to be considered separately. We extend our definition of homomorphisms to accommodate this kind of partial equivalence. This allows us to model *context dependent equivalences* as well. For example, for driving a nail, a shoe may be the best example of a hammer available, although these objects are not equivalent in general.

The minimization framework we develop for MDPs can be employed readily by RL algorithms for spatial abstraction in "flat" systems. The options framework (Sutton et al., 1999; Precup, 2000) enables RL algorithms to employ temporal abstractions in the form of temporally extended actions, or *options*. Options are composed of primitive actions are composed of primitive actions and take multiple time steps to execute, but for the purposes of problem solving they are considered as a single action. Thus we can think of *walk-to-the-door* as a single action, while in reality it is composed of a multitude of muscle twitches. Extending our algebraic framework to a hierarchical RL setting, such as the options framework, opens up additional possibilities.

We introduce *relativized options*, an extension to the option framework based on partial homomorphisms that allows us to define option policies without an absolute frame of reference. This widens the applicability of an option and also enables more efficient knowledge transfer across tasks and more efficient use of experience. Relativized options are related to the notion of *relativized operators* introduced by Iba (1989) as a compact representation of a family of related macro-operators. We also investigate the use of relativized options under approximate abstractions and in complex domains.

Options introduce new "behaviors" that enable abstractions that were not possible earlier. For example, consider a robot whose mission involves grasping a cylinder. Let us define an option to grasp the cylinder starting from any state in which the robot is physically close to the cylinder. Without such an option the robot would need to execute a different set of actions to grasp the cylinder from different starting states. Now it just needs to use the *grasp-cylinder* option. Therefore we can, under suitable circumstances, consider all these state-option pairs as equivalent. We extend our abstraction framework to employ definitions of homomorphisms and symmetry groups over options which allow us to model such *option induced abstractions.*

Another interpretation of relativized options is as a framework for defining option schemas. Option schemas are abstract templates of how to respond to a given situation. We model an abstract template as a partial homomorphic image of the original problem. When an agent invokes a schema it appropriately allocates, or binds, various resources and sensory capabilities to make the schema relevant to the specific instance. We model this as choosing the appropriate homomorphism to apply in a given situation. We develop algorithms for learning the appropriate binding of resources and empirically demonstrate the utility of employing option schemas.

Problems set in environments with objects often exhibit various symmetries and considerable redundancy in the representation. One way to exploit such symmetry is by employing representations known as indexical or *deictic representations* (Agre, 1988). The world is sensed via multiple pointers and the actions are specified with respect to these pointers. In this work we show that in some cases employing such deictic pointers is equivalent to identifying homomorphic reductions, and we develop a principled deictic RL algorithm based on the relativized options framework.

We begin by providing some background regarding MDPs and RL in the next chapter. We also introduce the notation that we will be using. In Chapter 3 we introduce MDP homomorphisms and formulate the model minimization problem in terms of homomorphisms. We develop the basic minimization algorithm and establish the equivalence of MDP homomorphisms and stochastic bisimulations. In Chapter 4 we define symmetry groups of MDPs, and present methods that take advantage of symmetry and structure. We also introduce approximate homomorphisms based on relaxed minimization criteria and derive bounds in the loss of performance. Chapter 5 deals with several aspects of combining hierarchical RL and homomorphisms. We introduce relativized options and present empirical demonstration of their utility. We also explore the use of approximate homomorphisms in this setting. In Chapter 6 we introduce option schemas and develop one approach to employing deictic representations within our framework. We present experimental results in complex domains. We conclude in Chapter 7 by examining the import of this work and suggesting future directions of research.

CHAPTER 2

BACKGROUND AND NOTATION

In this chapter we introduce some notation that we will use in the thesis. We also provide some background on minimization approaches for various modeling paradigms and a limited introduction to reinforcement learning.

2.1 Markov Decision Processes

A finite *Markov decision process* is a tuple $\langle S, A, \Psi, P, R \rangle$, where S is the set of states, A is the set of actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $P : \Psi \times S \rightarrow [0, 1]$ is the transition probability function with $P(s, a, s')$ being the probability of transition from state s to state s' under action a, and $R : \Psi \rightarrow \rm I\!R$ is the expected reward function, with $R(s, a)$ being the expected reward for performing action a in state s. We assume that the rewards are bounded. Let $A_s = \{a|(s, a) \in \Psi\} \subseteq A$ denote the set of actions admissible in state s. We assume that for all $s \in S$, A_s is non-empty. In this work we assume that the set of states and set of actions are finite, but the language of homomorphisms we employ extends to infinite spaces with little work.

A *stochastic policy* π is a mapping from Ψ to the real interval $[0, 1]$ s.t. $\sum_{a \in A_s} \pi(s, a) = 1$ for all $s \in S$. For any $(s, a) \in \Psi$, $\pi(s, a)$ gives the probability of picking action a in state s. The *value* of state s under policy π is the expected value of the discounted sum of future rewards starting from state s and following policy π thereafter. The *value function* V^π corresponding to a policy π is the mapping from states to their values under π. It can be shown (e. g., Bertsekas, 1987) that V^π satisfies the *Bellman equation*:

$$V^\pi(s) = \sum_{a \in A_s} \pi(s,a) \left[R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^\pi(s') \right],$$

where $0 \leq \gamma < 1$ is a discount factor. This formulation is known as the discounted sum of rewards criterion.

Similarly, the value of a state-action pair (s,a) under policy π is the expected value of the discounted sum of future rewards starting from state s, taking action a, and following π thereafter. The *action value function* Q^π corresponding to a policy π is the mapping from state-action pairs to their values and satisfies:

$$Q^\pi(s,a) = R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V^\pi(s'),$$

where $0 \leq \gamma < 1$ is a discount factor.

The solution of an MDP is an *optimal policy* $\pi^\star$ that uniformly dominates all other possible policies for that MDP. In other words $V^{\pi^{star}}(s) \geq V^\pi(s)$ for all s in S and for all possible π. It can be shown (Bertsekas, 1987) that the value functions for all optimal policies is the same. We denote this *optimal value function* by $V^\star$. It satisfies the *Bellman optimality equation*:

$$V^\star(s) = \max_{a \in A_s} \sum_{s' \in S} P(s,a,s') \left[R(s,a) + \gamma V^\star(s') \right].$$

Similarly the *optimal action value function* $Q^\star$ satisfies:

$$Q^\star(s,a) = \sum_{s' \in S} P(s,a,s') \left[R(s,a) + \gamma \max_{a' \in A_{s'}} Q^\star(s',a') \right].$$

These two optimal value functions are related by $V^\star(s) = \max_a Q^\star(s,a)$. Typically MDPs are solved by approximating the solution to the Bellman optimality equations

(e. g., Bertsekas, 1987; Sutton and Barto, 1998). Given the optimal action value function, an optimal policy is given by

$$\begin{aligned} \pi^\star(s,a) &\geq 0 \quad \text{if } Q^\star(s,a) = \max_{a' \in A_s} Q^\star(s,a') \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

2.2 Semi-Markov Decision Processes

A finite discrete time semi-Markov decision process (SMDP) is a generalization of a finite MDP in which actions can take variable amounts of time to complete. As with an MDP, an SMDP is a tuple $\langle S, A, \Psi, P, R \rangle$, where S, A and Ψ are the sets of states, actions and admissible state-action pairs; $P : \Psi \times S \times \mathbb{N} \rightarrow [0,1]$ is the transition probability function with $P(s,a,s',N)$ being the probability of transition from state s to state s' under action a in N time steps, and $R : \Psi \times \mathbb{N} \rightarrow \mathbb{R}$ is the expected discounted reward function, with $R(s,a,N)$ being the expected reward for performing action a in state s and completing it in N time steps.[1] We are adopting the formalism of Dietterich (2000a). The traditional approach (Howard, 1960) is to use two distributions to describe the state transitions, one of which is the usual next state distribution of MDPs and the other is a distribution of *holding times*. The holding time distribution is usually a function of the current state and action alone. We agree with Dietterich that the joint distribution formulation is more useful in modeling various hierarchical learning architectures, some of which we introduce shortly.

2.3 Reinforcement Learning

Reinforcement learning (RL) (Sutton and Barto, 1998) refers to a collection of learning algorithms that seek to approximate solutions to stochastic sequential de-

[1]Here $\mathbb{N}$ denotes the set of natural numbers.

cision tasks with scalar evaluative feedback. RL algorithms are designed to operate online and in close interaction with the environment in which the agent is operating. When a stochastic sequential decision problem is modeled as an MDP, RL algorithms try to estimate the optimal value function and/or optimal policy.

Many of the popular RL algorithms are based on the Q-learning (Watkins, 1989) approach that seeks to approximate the optimal action value function through online experience. After experiencing a transition from state s to s' under action a and observing a reward of r, Q-learning employs the following update:

$$Q^\star(s,a) \leftarrow (1-\alpha)Q^\star(s,a) + \alpha\left[r + \gamma \max_{a'\in A(s')} Q^\star(s',a')\right]$$

where α is a learning rate between 0 and 1. It has been shown that under suitable conditions Q-learning converges to the optimal action value function (Watkins and Dayan, 1992).

Bradtke and Duff (1995) introduced a straightforward extension of Q-learning for continuous time SMDPs, known as SMDP Q-learning. In the discrete time case, after experiencing a transition from state s to s' in k time steps under action a and observing a sequence of rewards $r_1, \cdots, r_k$, SMDP Q-learning employs the following update:

$$Q^\star(s,a) \leftarrow (1-\alpha)Q^\star(s,a) + \alpha\left[r + \gamma^k \max_{a'\in A(s')} Q^\star(s',a')\right]$$

where $r = \sum_{j=0}^{k-1} \gamma^j r_{j+1}$ is the *discounted return* and α is a learning rate between 0 and 1. It has been shown that under the same conditions as Q-learning, SMDP Q-learning converges to the optimal action value function (Parr, 1998).

2.4 Partitions, maps and equivalence relations

A *partition* B of a set X is a collection of disjoint subsets, or *blocks*, $b_i \subseteq X$ such that $\bigcup_i b_i = X$. For any $x \in X$, $[x]_B$ denotes the block of B to which x belongs. Let

B_1 and B_2 be partitions of X. We say that B_1 is *coarser than* B_2 (or B_2 is a *refinement of* B_1), denoted $B_1 \geq B_2$, if for all $x, x' \in X$, $[x]_{B_2} = [x']_{B_2}$ implies $[x]_{B_1} = [x']_{B_1}$. The relation $\geq$ is a partial order on the set of partitions of X.

To any partition B of X there corresponds an equivalence relation, $\equiv_B$, on X with $x \equiv_B x'$ if and only if $[x]_B = [x']_B$ for all $x, x' \in X$. Any function f from a set X into a set Y defines an equivalence relation on X with $x \equiv_f x'$ if and only if $f(x) = f(x')$. We say that x and x' are *f-equivalent* when $x \equiv_f x'$, and we denote the partition of X corresponding to this equivalence relation by B_f.

Let B be a partition of $Z \subseteq X \times Y$, where X and Y are arbitrary sets. For any $x \in X$, let $B(x)$ denote the set of distinct blocks of B containing pairs of which x is a component, that is, $B(x) = \{[(w, y)]_B \mid (w, y) \in Z, w = x\}$. The *projection of B onto X* is the partition $B|X$ of X such that for any $x, x' \in X$, $[x]_{B|X} = [x']_{B|X}$ if and only if $B(x) = B(x')$. In other words, $x \equiv_{B|X} x'$ if and only if every block of B containing a pair in which x (x') is a component also contains a pair in which x' (x) is a component.[2] Note that if B_1 and B_2 are partitions of Z, then $B_1 \geq B_2$ implies that $B_1|X \geq B_2|X$.

A *partition of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$* is a partition of Ψ. Given a partition B of $\mathcal{M}$, the *block transition probability of $\mathcal{M}$* is the function $T : \Psi \times B|S \to [0, 1]$ defined by $T(s, a, [s']_{B|S}) = \sum_{s'' \in [s']_{B|S}} P(s, a, s'')$. In other words, when applying action a in state s, $T(s, a, [s']_{B|S})$ is the probability that the resulting state is in the block $[s']_{B|S}$. It is clear that since $B|S$ is a partition of S, each of these block transition probabilities is in the interval $[0, 1]$.

[2]The more traditional definition of a projection is: $x \equiv_{B|X} x'$ if and only if $(x, y) \equiv_B (x', y)$ for all $y \in Y$. This projection is always a refinement of the our projection. We need the modified definition to facilitate the development of some concepts below.

Example 1

Let $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ be an MDP with $S = \{s_1, s_2, s_3\}$, $A = \{a_1, a_2\}$ and $\Psi = \{(s_1, a_1), (s_1, a_2), (s_2, a_1), (s_2, a_2), (s_3, a_1)\}$. We give the projections under both our definition and the traditional one (see footnote). The traditional projection does not lead to aggregation of states in any of the cases, while our definition does in the first two cases. In the last case both definitions result in singletons.

i) If $B_1 = \Big\{\{(s_1, a_1), (s_2, a_2)\}, \{(s_1, a_2), (s_2, a_1), (s_3, a_1)\}\Big\}$,
then $B_1|S = \Big\{\{s_1, s_2\},\ \{s_3\}\Big\}$ (ours); $\Big\{\{s_1\},\ \{s_2\}, \{s_3\}\Big\}$ (traditional).

ii) If $B_2 = \Big\{\{(s_2, a_1)\}, \{(s_1, a_1), (s_1, a_2), (s_2, a_2), (s_3, a_1)\}\Big\}$,
then $B_2|S = \Big\{\{s_1, s_3\},\ \{s_2\}\Big\}$; $\quad \Big\{\{s_1\},\ \{s_2\}, \{s_3\}\Big\}$.

iii) If $B_3 = \Big\{\{(s_1, a_1), (s_2, a_2)\}, \{(s_1, a_2), (s_3, a_1)\}, \{(s_2, a_1)\}\Big\}$,
then $B_3|S = \Big\{\{s_1\},\ \{s_2\}, \{s_3\}\Big\}$; $\quad \Big\{\{s_1\},\ \{s_2\}, \{s_3\}\Big\}$.

□

CHAPTER 3

MDP HOMOMORPHISMS AND MINIMIZATION

In this chapter we develop the mathematical formulation that underlies our approach to abstraction. In particular we want a notion of equivalence among state-action pairs that can capture the various intuitive notions of redundancy and similarity, such as aggregate representations, symmetries, object replacement etc. The notion we adopt is that of a *MDP homomorphism*.

In order to be able to model a wide class of abstractions, we introduce a broad notion of equivalence under which two states are considered equivalent if for every action admissible in one state there is some action, not necessarily the same, admissible in the other state that produces similar results. Earlier notions of equivalence for MDPs required that the same action produce similar results in both states. Referring back to Figure 1.2(a), states A and B are considered equivalent since for every action from A there is an equivalent, though different, action in B. We characterize our notion of equivalence by certain conditions on the transition probabilities and the expected immediate rewards. While many mathematical formalisms can be employed here, we choose to extend the notion of machine homomorphisms from the FSA literature. We develop MDP homomorphisms starting from a simple case working our way up.

3.1 Group Homomorphism

Informally, a homomorphism of a structured system is some transformation of the system that preserves aspects of this structure. One simple structured mathematical concept is a *group*. A group G is a set, together with an operator, denoted $+$. This

$$\begin{tikzcd}
G \times G \arrow[r, "+"] \arrow[d, "f"'] & G \arrow[d, "f"] \\
G' \times G' \arrow[r, "+'"] & G'
\end{tikzcd}$$

Figure 3.1. A Group Homomorphism represented by Commutative Diagrams

associates with each pair of elements x and y of G another element $x + y$ in G. A group satisfies the properties of *associativity* ($(x + y) + z = x + (y + z)$), existence of an identity e ($x + e = e + x = x$), and the existence of an inverse x^{-1} for each element x ($x + x^{-1} = x^{-1} + x = e$). The set of integers with the operation of addition is an example of a group, known as the additive group of integers.

Let G and G' be two groups. A homomorphism, f, is a map from G to G' having the following property, for all $x, y \in G$:

$$f(x + y) = f(x) +' f(y)$$

The map f is said to *commute* with the group operator $+$. We can start with two elements in G apply the operator and then f or we can apply f to each element individually and then apply the operator in G' and we end up the same result in G'. This is illustrated in the *commutative diagram* shown in Figure 3.1. As an example, consider the set of even integers. This is a group under addition. The function $f(x) = 2x$ from the additive group of integers to the additive group of even integers is a homomorphism, since $f(x + y) = 2(x + y) = 2x + 2y = f(x) + f(y)$.

3.2 Machine Homomorphism

The finite state automata (FSA) literature is rich in algebraic approaches to minimizing and decomposing machines. Most approaches are based on the concept of machine homomorphism and notions of equivalence (of states and of machines) derived from it. This thesis extends the concept of machine homomorphism to an MDP homomorphism and develops similar notions of equivalence applicable in RL. In the case of FSA we want the homomorphism to preserve the transition behavior and output characteristics of automata.

Formally, an FSA is given by $\mathcal{F} = \langle S, s_0, \Sigma, Z, \delta, O \rangle$, where S is the set of states, s_0 is the start state, Σ the set of input symbols, Z the set of output symbols, $\delta : S \times \Sigma \rightarrow S$ the transition function and $O : S \rightarrow Z$ the output function. A machine (FSA) homomorphism from $\mathcal{F} = \langle S, s_0, \Sigma, Z, \delta, O \rangle$ to $\mathcal{F}' = \langle S', s_0', \Sigma, Z, \delta', O' \rangle$ is a surjection f from S to S' such that $f(\delta(s, \sigma)) = \delta'(f(s), \sigma)$ and $O(s) = O'(f(s))$.[1] The homomorphism f is said to *commute* with the dynamics and *respect* the output function of $\mathcal{F}$. We can depict this using commutative diagrams as shown in Figure 3.2. Here horizontal and diagonal arrows represent system dynamics and vertical arrows represent homomorphisms. Starting from a particular element in S, regardless of the pair of arrows we follow we end up with the same element in S'. Similarly, the second diagram illustrates the commutative property for the output Z. A machine homomorphism is also known as a *dynamorphism* in category theory (Arbib and Manes, 1975).

The homomorphism f is only a surjection and often S' is much smaller than S. In such cases, we can construct a reduced model of an FSA from the partition of the state space induced by f. This reduced model will be equivalent to $\mathcal{F}'$ up to a

[1]If f is not a surjection then there exists a closed sub-machine of F' that is a homomorphic image of F and we consider this sub-machine as the image of the homomorphism. Such a map f is also known as a *simulation.*

$$\begin{tikzcd}
s_1 \arrow[r, mapsto, "\delta(\cdot,\sigma)"] \arrow[d, mapsto, "f"'] & s_2 \arrow[d, mapsto, "f"] \\
s_1' \arrow[r, mapsto, "\delta'(\cdot,\sigma)"] & s_2'
\end{tikzcd}
\qquad
\begin{tikzcd}
s_1 \arrow[r, mapsto, "O(\cdot)"] \arrow[d, mapsto, "f"'] & z \\
s_1' \arrow[ur, mapsto, "O'(\cdot)"] &
\end{tikzcd}$$

Figure 3.2. An FSA homomorphism represented by commutative diagrams.

relabeling of states and outputs and would have the same "block" transition behavior as $\mathcal{F}$.

3.3 MDP Homomorphism

We extend the notion of machine homomorphisms to MDPs by incorporating stochasticity, decision making and rewards. An MDP homomorphism is a map on Ψ that commutes with the system dynamics and preserves the reward structure. Formally, we define it as:

Definition: An *MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$, with $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that for all $s, t \in S$, and $a \in A_s$:

$$\begin{aligned} P'(f(s), g_s(a), f(t)) &= T(s, a, [t']_{B_h|S}), & (3.1) \\ R'(f(s), g_s(a)) &= R(s, a). & (3.2) \end{aligned}$$

We call $\mathcal{M}'$ the *homomorphic image* of $\mathcal{M}$ under h. We use the shorthand $h(s,a)$ to denote $h((s,a))$. The surjection f maps states of $\mathcal{M}$ to states of $\mathcal{M}'$, and since it is generally many-to-one, it generally induces nontrivial equivalence classes of states s of M: $[s]_f$. Each surjection g_s recodes the actions admissible in state s of $\mathcal{M}$ to

$$\begin{tikzcd}
(s,a) \arrow[r, mapsto, "P"] \arrow[d, mapsto, "h"'] & P_{sa} \arrow[d, mapsto, "h"] \\
(s',a') \arrow[r, mapsto, "P'"] & P'_{s'a'}
\end{tikzcd}
\qquad
\begin{tikzcd}
(s,a) \arrow[r, mapsto, "R"] \arrow[d, mapsto, "h"'] & \overline{r} \\
(s',a') \arrow[ur, mapsto, "R'"] &
\end{tikzcd}$$

Figure 3.3. An MDP Homomorphism represented by Commutative Diagrams

actions admissible in state $f(s)$ of $\mathcal{M}'$. This *state-dependent* recoding of actions is a key innovation of our definition, which we discuss in more detail below. Condition (1) says that the transition probabilities in the simpler MDP $\mathcal{M}'$ are expressible as sums of the transition probabilities of the states of $\mathcal{M}$ that f maps to that same state in $\mathcal{M}'$. This is the stochastic version of the standard condition for homomorphisms of deterministic systems that requires that the homomorphism commutes with the system dynamics (Hartmanis and Stearns, 1966). Condition (2) says that state-action pairs that have the same image under h have the same expected reward.

Let $P_{sa} : S \rightarrow [0,1]$ be the distribution over states resulting from taking action a in state s, i.e., $P_{sa}(t) = P(s,a,t)$ for any t in S. The *aggregation* hP_{sa}, of P_{sa} over the homomorphism h, is the distribution over S' such that $hP_{sa}(s') = \sum_{t \in f^{-1}(s')} P_{sa}(t)$ for each $s' \in S'$. Here $f^{-1}(s') = \{s \in S | f(s) = s'\}$ is the pre-image of s' in S. A homomorphism commutes with the one step dynamics of the MDP in the sense that the aggregation hP_{sa} is the same distribution as $P'_{f(s)g_s(a)}$ for all $(s,a) \in \Psi$. We can depict this using commutative diagrams shown in Figure 3.3.

MDP homomorphisms lead to the following notions of equivalence of states and state-action pairs which, as shown in the next section, lead to the intuitive notion of equivalence we are interested in modeling.

Definition: State action pairs (s_1, a_1) and $(s_2, a_2) \in \Psi$ are (homomorphically) *equivalent* if for some homomorphism h of $\mathcal{M}$, $(s_1, a_1) \equiv_h (s_2, a_2)$.

Definition: States s_1 and $s_2 \in S$ are *equivalent* if i) for every action $a_1 \in A_{s_1}$, there is an action $a_2 \in A_{s_2}$ such that (s_1, a_1) and (s_2, a_2) are equivalent, and ii) for every action $a_2 \in A_{s_2}$, there is an action $a_1 \in A_{s_1}$, such that (s_1, a_1) and (s_2, a_2) are equivalent.

Thus the surjection f maps equivalent states of $\mathcal{M}$ onto the same image state in $\mathcal{M}'$, while g_s is a *state dependent* mapping of the actions in $\mathcal{M}$ onto image actions in $\mathcal{M}'$. For example, if $h = \langle f, \{g_s | s \in S\} \rangle$ is a homomorphism from the gridworld of Figure 1.2(a) to that of Figure 1.2(b), then $f(A) = f(B)$ is the state marked $\{A, B\}$ in Figure 1.2(b). Also $g_A(E) = g_B(N) = E$, $g_A(W) = g_B(S) = W$, and so on.

The rest of the chapter, and in some sense the rest of the thesis, focuses on answering the questions: How useful is this formulation of MDP homomorphisms? Is it general enough to model a wide class of abstractions? Is it powerful enough to result in computational savings? Is it well-defined—is the optimality of solutions preserved? We start by looking at the last question in detail.

3.4 Minimization Framework

Our approach to abstraction can be considered an instance of a general approach known as *model minimization.* The goal of MDP minimization is to form a reduced model of a system by ignoring irrelevant information. Solving this reduced model should then yield a solution to the original MDP. Frequently minimization is accomplished by identifying states and actions that are equivalent in a well-defined sense and forming a "quotient" model by aggregating such states and actions. We build a minimization framework, that is an extension of a framework by Dean and Givan

(1997). It differs from their work in the notion of equivalence we employ based on MDP homomorphisms.

In this section we show that homomorphic equivalence leads to preservation of optimal solutions. We start with the following theorem on optimal value equivalence. This theorem is an extension of the optimal value equivalence theorem developed in Givan et al. (2003) for stochastic bisimulations.

Theorem 1: (*Optimal value equivalence*) Let $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ be the homomorphic image of the MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ under the MDP homomorphism $h = \langle f, \{g_s | s \in S\} \rangle$. For any $(s, a) \in \Psi$, $Q^\star(s, a) = Q^\star(f(s), g_s(a))$.

Proof: (Along the lines of Givan et al. (2003)) Let us define the m-step optimal discounted action value function recursively for all $(s, a) \in \Psi$ and for all non-negative integers m as

$$Q_m(s, a) = R(s, a) + \gamma \sum_{s_1 \in S} \left[P(s, a, s_1) \max_{a_1 \in A_{s_1}} Q_{m-1}(s_1, a_1) \right]$$

and set $Q_{-1}(s_1, a_1) = 0$. Letting $V_m(s_1) = \max_{a_1 \in A_{s_1}} Q_m(s_1, a_1)$, we can rewrite this as:

$$Q_m(s, a) = R(s, a) + \gamma \sum_{s_1 \in S} \left[P(s, a, s_1) V_{m-1}(s_1) \right].$$

Now we prove by induction on m that the theorem is true. For the base case of $m = 0$, we have that $Q_0(s, a) = R(s, a) = R'(f(s), g_s(a)) = Q_0(f(s), g_s(a))$. Now let us assume that $Q_j(s, a) = Q_j(f(s), g_s(a))$ for all values of j less than m and all state-action pairs in Ψ. Now we have,

$$\begin{aligned}
Q_m(s, a) &= R(s, a) + \gamma \sum_{s' \in S} P(s, a, s') V_{m-1}(s') \\
&= R(s, a) + \gamma \sum_{[s']_{B_h|S} \in B_h|S} T(s, a, [s']_{B_h|S}) V_{m-1}(s') \\
&= R'(f(s), g_s(a)) + \gamma \sum_{s' \in S'} P'(f(s), g_s(a), s') V_{m-1}(s')
\end{aligned}$$

$$= \quad Q_m(f(s), g_s(a))$$

The second and third lines use the fact that h is a homomorphism. Since R is bounded it follows by induction that $Q^\star(s, a) = Q^\star(f(s), g_s(a))$ for all $(s, a) \in \Psi$. □

Corollaries:

1. For any h-equivalent $(s_1, a_1), (s_2, a_2) \in \Psi$, $Q^\star(s_1, a_1) = Q^\star(s_2, a_2)$.

2. For all equivalent $s_1, s_2 \in S$, $V^\star(s_1) = V^\star(s_2)$.

3. For all $s \in S$, $V^\star(s) = V^\star(f(s))$.

Proof: Corollary 1 follows from Theorem 1. Corollaries 2 and 3 follow from Theorem 1 and the fact that $V^\star(s) = \max_{a \in A_s} Q^\star(s, a)$. □

As shown by Givan et al. (2003), optimal value equivalence is not a sufficient notion of equivalence for our stated minimization goal. In many cases even when the optimal values are equal, the optimal policies might not be related and hence we cannot easily transform solutions of $\mathcal{M}'$ to solutions of $\mathcal{M}$. But when $\mathcal{M}'$ is a homomorphic image, a policy in $\mathcal{M}'$ can *induce* a policy in $\mathcal{M}$ that is closely related. The following describes how to derive such an induced policy.

Definition: Let $\mathcal{M}'$ be an image of $\mathcal{M}$ under homomorphism $h = \langle f, \{g_s | s \in S\}\rangle$. For any $s \in S$, $g_s^{-1}(a')$ denotes the set of actions that have the same image $a' \in A'_{f(s)}$ under g_s. Let π' be a stochastic policy in $\mathcal{M}'$. Then π' *lifted to* $\mathcal{M}$ is the policy $\pi'_{\mathcal{M}}$ such that for any $a \in g_s^{-1}(a')$, $\pi'_{\mathcal{M}}(s, a) = \pi'(f(s), a') \Big/ |g_s^{-1}(a')|$.
Note: It is sufficient that $\sum_{a \in g_s^{-1}(a')} \pi'_{\mathcal{M}}(s, a) = \pi'(f(s), a')$, but we use the above definition to make the lifted policy unique.

Example 2

This example illustrates the process of lifting a policy from an image MDP to the original MDP. Consider MDP $\mathcal{M}$ from example 1 and $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ with

$S' = \{s'_1, s'_2\}$, $A' = \{a'_1, a'_2\}$ and $\Psi' = \{(s'_1, a'_1),\ (s'_1, a'_2),\ (s'_2, a'_1)\}$. Let $h = \langle f, \{g_s | s \in S\}\rangle$ be a homomorphism from $\mathcal{M}$ to $\mathcal{M}'$ defined by

$$\begin{array}{lll} f(s_1) = s'_1 & f(s_2) = s'_2 & f(s_3) = s'_2 \\ g_{s_1}(a_1) = a'_2 & g_{s_2}(a_1) = a'_1 & g_{s_3}(a_1) = a'_1 \\ g_{s_1}(a_2) = a'_1 & g_{s_2}(a_2) = a'_1 & \end{array}$$

Let π' be a policy in $\mathcal{M}'$ with

$$\pi'(s'_1, a'_1) = 0.6 \qquad \pi'(s'_1, a'_2) = 0.4 \qquad \pi'(s'_2, a'_1) = 1.0$$

Now π' lifted to $\mathcal{M}$, the policy $\pi'_{\mathcal{M}}$, is derived as follows:

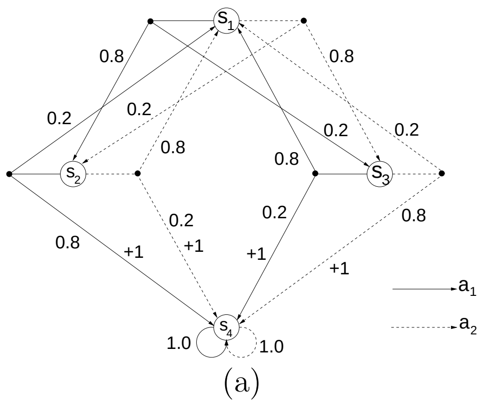
$$\begin{array}{ll} \pi'_{\mathcal{M}}(s_1, a_1) = \pi'(s'_1, a'_2) = 0.4 & \pi'_{\mathcal{M}}(s_1, a_2) = \pi'(s'_1, a'_1) = 0.6 \\ \pi'_{\mathcal{M}}(s_2, a_1) = \pi'(s'_2, a'_1)/2 = 0.5 & \pi'_{\mathcal{M}}(s_2, a_2) = \pi'(s'_2, a'_1)/2 = 0.5 \\ \pi'_{\mathcal{M}}(s_3, a_1) = \pi'(s'_2, a'_1) = 1.0 & \end{array}$$

□

Theorem 2: Let $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ be the image of $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ under the homomorphism $h = \langle f, \{g_s | s \in S\}\rangle$. If $\pi'^{\star}$ is an optimal policy for $\mathcal{M}'$, then $\pi'^{\star}_{\mathcal{M}}$ is an optimal policy for $\mathcal{M}$.

Proof: Let $\pi'^{\star}$ be an optimal policy in $\mathcal{M}'$. Consider some $(s, a) \in \Psi$ such that $\pi'^{\star}(f(s), g_{s_1}(a_1))$ is greater than zero. Then $Q^{\star}(f(s_1), g_{s_1}(a_1))$ is the maximum value of the $Q^{\star}$ function in state $f(s_1)$. From Theorem 1, we know that $Q^{\star}(s, a) = Q^{\star}(f(s), g_s(a))$ for all $(s, a) \in \Psi$. Therefore $Q^{\star}(s_1, a_1)$ is the maximum value of the $Q^{\star}$ function in state s_1. Thus a_1 is an optimal action in state s_1 and hence $\pi'^{\star}_{\mathcal{M}}$ is an optimal policy for $\mathcal{M}$. □

Theorem 2 establishes that an MDP can be solved by solving one of its homomorphic images. To achieve the most impact, we need to derive a smallest homomorphic



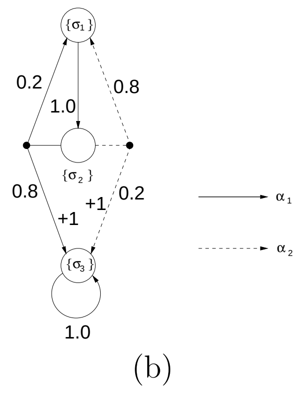


Figure 3.4. (a) Transition graph of example MDP $\mathcal{M}$. This MDP is irreducible under a traditional minimization framework. Our notion of homomorphic equivalence allows us to minimize this further. (b) Transition graph of the minimal image of the MDP $\mathcal{M}$ in (a).

image of the MDP, i.e., an image with the least number of admissible state-action pairs. The following definition formalizes this notion.

Definition: An MDP $\mathcal{M}$ is a *minimal MDP* if for every homomorphic image $\mathcal{M}'$ of $\mathcal{M}$, there exists a homomorphism from $\mathcal{M}'$ to $\mathcal{M}$. A *minimal image* of an MDP $\mathcal{M}$ is a homomorphic image of $\mathcal{M}$ that is also a minimal MDP.

The model minimization problem can now be stated as: "find a minimal image of a given MDP". Since this can be computationally prohibitive, we frequently settle for a reasonably reduced model, even if it is not a minimal MDP.

Illustration of Minimization: An Abstract MDP example

We illustrate our minimization framework on a very simple abstract MDP shown in Figure 3.4(a). We will use this as a running example while we develop our framework further. We chose such a simple example in order to make the presentation of the computation involved in later stages easier. Note though that this MDP is irreducible under the state-equivalence based MDP minimization framework of Dean and Givan. The parameters of $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ are $S = \{s_1, s_2, s_3, s_4\}$, $A = \{a_1, a_2\}$, $\Psi =$

$S \times A$, P defined as in Table 3.1 and R given by: $R(s_2, a_1) = R(s_3, a_2) = 0.8$ and $R(s_2, a_2) = R(s_3, a_1) = 0.2$. For all other values of i and j, $R(s_i, a_j)$ equals zero.

to → ↓ from	s_1	s_2	s_3	s_4
s_1	0	0.8	0.2	0
s_2	0.2	0	0	0.8
s_3	0.8	0	0	0.2
s_4	0	0	0	1.0

(i)

to → ↓ from	s_1	s_2	s_3	s_4
s_1	0	0.2	0.8	0
s_2	0.8	0	0	0.2
s_3	0.2	0	0	0.8
s_4	0	0	0	1.0

(ii)

Table 3.1. Transition probabilities for the MDP $\mathcal{M}$ shown in Figure 3.4(a): (i) under action a_1. (ii) under action a_2.

The MDP $\mathcal{M}'$ shown in Figure 3.4(b) is a homomorphic image of $\mathcal{M}$. It has the following parameters: $S' = \{\sigma_1, \sigma_2, \sigma_3\}$, $A' = \{\alpha_1, \alpha_2\}$, $\Psi' = \{(\sigma_1, \alpha_1), (\sigma_2, \alpha_1), (\sigma_2, \alpha_2), (\sigma_3, \alpha_1)\}$, P' as shown in Table 3.2 and R' defined as follows: $R'(\sigma_2, \alpha_1) = 0.2$, $R'(\sigma_2, \alpha_2) = 0.8$ and all other rewards are zero.

$$\begin{array}{ll} P'(\sigma_1, \alpha_1, \sigma_2) = 1.0 & P'(\sigma_3, \alpha_1, \sigma_3) = 1.0 \\ P'(\sigma_2, \alpha_1, \sigma_1) = 0.8 & P'(\sigma_2, \alpha_2, \sigma_1) = 0.2 \\ P'(\sigma_2, \alpha_1, \sigma_3) = 0.2 & P'(\sigma_2, \alpha_2, \sigma_3) = 0.8 \end{array}$$

Table 3.2. The transition probabilities of the MDP $\mathcal{M}'$ shown in Figure 3.4(b).

One can define a homomorphism $\langle f, \{g_s | s \in S\}\rangle$ from $\mathcal{M}$ to $\mathcal{M}'$ as follows: $f(s_1) = \sigma_1$, $f(s_2) = f(s_3) = \sigma_2$, and $f(s_4) = \sigma_3$. $g_{s_1}(a_i) = g_{s_4}(a_i) = \alpha_1$, for $i = 1, 2$, $g_{s_2}(a_1) = g_{s_3}(a_2) = \alpha_2$ and $g_{s_2}(a_2) = g_{s_3}(a_1) = \alpha_1$.

3.5 Identifying Homomorphisms

Now that we have established that homomorphic equivalence is useful in deriving reduced models, the natural next question is how do we identify homomorphisms? Is it computationally feasible? In this section we develop the basic minimization approach and introduce a simple minimization algorithm. This algorithm takes time polynomial in the size of Ψ.

A homomorphism from $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ induces a partition on Ψ. Classical FSA literature employs such partitions of the state set in minimization of machines. Likewise, one approach to constructing a suitable image MDP is to identify partitions of Ψ that correspond to equivalence classes of homomorphisms. For that we first need to establish conditions under which a partition corresponds to a homomorphism.

Definition: A partition B of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ is said to be *reward respecting* if $B_R \geq B$.[2] In other words B is reward respecting if $(s_1, a_1) \equiv_B (s_2, a_2)$ implies $R(s_1, a_1) = R(s_2, a_2)$ for all $(s_1, a_1), (s_2, a_2) \in \Psi$.

Definition: A partition B of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ has the *stochastic substitution property* if for all $(s_1, a_1), (s_2, a_2) \in \Psi$, $(s_1, a_1) \equiv_B (s_2, a_2)$ implies $T(s_1, a_1, [s]_{B|S})$ $= T(s_2, a_2, [s]_{B|S})$ for all $[s]_{B|S} \in B|S$.

In other words, the block transition probability is the same for all state-action pairs in a given block. A partition that satisfies the stochastic substitution property is an *SSP partition.* This is an extension of the substitution property for finite state machines (Hartmanis and Stearns, 1966). The *SSP block transition probability* is the function $T_b : B \times B|S \to [0, 1]$, defined by $T_b([(s_1, a_1)]_B, [s]_{B|S}) = T(s_1, a_1, [s]_{B|S})$. This quantity is well-defined only for SSP partitions.

[2]Recall, B_R is the partition of Ψ induced by the reward function.

Theorem 3: Let h be an MDP homomorphism from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$. Then B_h, the partition of Ψ induced by h, is a reward respecting SSP partition.

Proof: Let $h = \langle f, \{g_s | s \in S\}\rangle$ be the homomorphism from $\mathcal{M}$ to $\mathcal{M}'$. We need to show that the partition B_h is a reward respecting SSP partition.

First let us tackle the stochastic substitution property. Let $(s_1, a_1), (s_2, a_2) \in \Psi$, be h-equivalent. From the definition of a homomorphism we have that $f(s_1) = f(s_2) = s' \in S'$ and $g_{s_1}(a_1) = g_{s_2}(a_2) = a' \in A'_{s'}$. Thus, for any $s \in S$, $T(s_1, a_1, [s]_{B_h|S}) = P'(s', a', f(s)) = T(s_2, a_2, [s]_{B_h|S})$. Hence B_h is an SSP partition.

From condition 2 in the definition of a homomorphism, it is clear that the partition induced is reward respecting. $\square$

Theorem 3 establishes that the partition induced by a homomorphism is a reward respecting SSP partition. On the other hand, given any reward respecting SSP partition B of $\mathcal{M}$ it is possible to construct a homomorphic image. Let $\eta(s)$ be the number of distinct classes of B that contain a state-action pair with s as the state component, and let $\{[(s, a_i)]_B \,|i = 1, 2, \cdots, \eta(s)\}$ be the blocks. Note that if $[s_1]_{B|S} = [s_2]_{B|S}$ then $\eta(s_1) = \eta(s_2)$, hence the following is well-defined.

Definition: Given a reward respecting SSP partition B of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$, the *quotient MDP* $\mathcal{M}/B$ is the MDP $\langle S', A', \Psi', P', R'\rangle$, where $S' = B|S$; $A' = \bigcup_{[s]_{B|S} \in S'} A'_{[s]_{B|S}}$ where $A'_{[s]_{B|S}} = \{a'_1, a'_2, \cdots, a'_{\eta(s)}\}$ for each $[s]_{B|S} \in S'$; P' is given by $P'([s]_f\,, a'_i, [s']_f) = T_b([(s, a_i)]_B\,, [s']_{B|S})$ and R' is given by $R'([s]_{B|S}\,, a'_i) = R(s, a_i)$.

Theorem 4: Let B be a reward respecting SSP partition of MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$. The quotient MDP $\mathcal{M}/B$ is a homomorphic image of $\mathcal{M}$.

Proof: Given a reward respecting SSP partition B of $\mathcal{M}$, we show by construction that there exists a homomorphism h from $\mathcal{M}$ to the quotient MDP $\mathcal{M}/B = \langle S', A', \Psi', P', R' \rangle$.

The homomorphism $h = \langle f, \{g_s | s \in S\} \rangle$ between $\mathcal{M}$ and $\mathcal{M}/B$ is given by $f(s) = [s]_{B|S}$ and $g_s(a) = a'_i$ such that $T(s, a, [s']_{B|S}) = P'([s]_{B|S}, a'_i, [s']_{B|S})$ for all $[s']_{B|S} \in B|S$. In other words, if $[(s, a)]_{B|S}$ is the i-th unique block in the ordering used in the construction of $\mathcal{M}/B$, then $g_s(a) = a'_i$. It is easy to verify that h is indeed a homomorphism. □

The partition induced on $\mathcal{M}$ by h is only guaranteed to be a refinement of B and is not always the same partition as B. In other words, $B \geq B_h$. In fact, B_h is the least coarse partition such that $B_h|S = B|S$, and $\mathcal{M}/B$ is the same MDP as $\mathcal{M}/B_h$ up to a relabeling of states and actions. Thus the converse of the theorem, that for every reward respecting SSP partition there exists a homomorphism that induces it, is not always true.

It is easy to verify (by contradiction) that there exists a *unique* coarsest reward respecting SSP partition for any MDP. Intuitively one would expect the quotient MDP corresponding to the coarsest reward respecting SSP partition of an MDP $\mathcal{M}$ to be a minimal image of $\mathcal{M}$. The following theorem states that formally.

Theorem 5: Let B be the coarsest reward respecting SSP partition of MDP $\mathcal{M}$. The quotient MDP $\mathcal{M}/B$ is a minimal image of $\mathcal{M}$.

Proof: The proof of this theorem is given in Appendix A.

Dean and Givan (1997) propose a polynomial time method to identify the coarsest reward respecting SSP partition of an MDP. Though their method operates with partitions on the state space only, it can easily be extended to Ψ. Given an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$, the outline of a basic model-minimization algorithm is as follows:

1. Start with any reward respecting partition B of Ψ. The most obvious choice is to pick the one that is induced by the expected reward function R. This is the coarsest possible reward respecting partition, but any suitable partition will do.
2. Pick some block b_i of B that does not satisfy the SSP property and split b_i so that it does.
3. Repeat step 2 until all violations of the SSP property are resolved. Let B_h be the resulting partition.
4. Form the quotient MDP $\mathcal{M}/B_h$ and identify the homomorphism between $\mathcal{M}$ and $\mathcal{M}/B_h$.

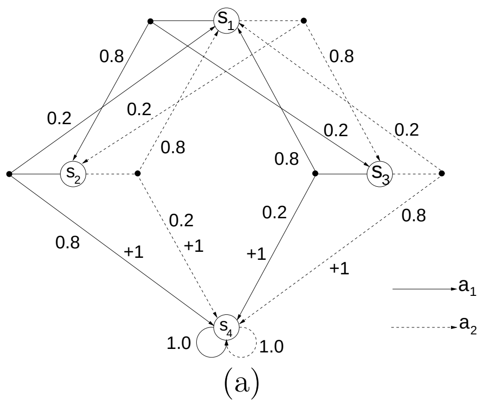
Now one can solve $\mathcal{M}/B_h$ and lift the optimal policy to get an optimal policy for $\mathcal{M}$. It can be shown (Dean and Givan, 1997) that step 2 has to be performed only once for each block in the partition and hence the algorithm runs in time quadratic in $|B_h|$ and linear in $|\Psi|$. The algorithm converges to the coarsest reward respecting SSP partition, provided we started with a suitable reward respecting partition.

Illustration of Minimization: An Abstract MDP example (revisited)

Let us return to the abstract MDP $\mathcal{M}$ from Figure 3.4(a), reproduced here in 3.5(a). We now derive the minimal model of this MDP. The admissible state action pairs is given by $S \times A$. We start with the partition induced by the reward function:

$$B_R = \Big\{\{(s_2, a_1), (s_3, a_2)\}, \{(s_2, a_2), (s_3, a_1)\}, \{(s_1, a_1), (s_1, a_2), (s_4, a_1), (s_4, a_2)\}\Big\}.$$

We denote the the blocks of the partition by b_1, b_2 and b_3 respectively. Now $B_R|S = \Big\{\{s_1, s_4\}, \{s_2, s_3\}\Big\}$, $T_b(b_1, \{s_1, s_4\}) = T_b(b_2, \{s_1, s_4\}) = 1.0$ and $T_b(b_1, \{s_2, s_3\}) =$



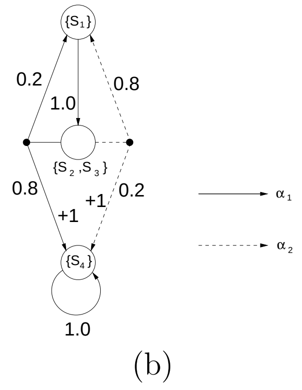


Figure 3.5. (a) Transition graph of example MDP $\mathcal{M}$. Repeated from Figure 3.4(a). (b) Transition graph of the quotient MDP $\mathcal{M}|B$. See text for description. Note that this is *isomorphic* to the MDP in Figure 3.4(b).

$T_b(b_2, \{s_2, s_3\}) = 0.0$. Hence b_1 and b_2 satisfy the SSP property and do not need to be split. Block b_3 does violate the SSP property as can be seen below:

$$
\begin{aligned}
T(s_1, a_1, \{s_1, s_4\}) &= 0 \qquad & T(s_4, a_1, \{s_1, s_4\}) &= 1.0 \\
T(s_1, a_2, \{s_1, s_4\}) &= 0 \qquad & T(s_4, a_2, \{s_1, s_4\}) &= 1.0 \\
T(s_1, a_1, \{s_2, s_3\}) &= 1.0 \qquad & T(s_4, a_1, \{s_2, s_3\}) &= 0 \\
T(s_1, a_2, \{s_2, s_3\}) &= 1.0 \qquad & T(s_4, a_2, \{s_2, s_3\}) &= 0
\end{aligned}
$$

We can fix this by splitting b_3 into $\Big\{\{(s_1, a_1), (s_1, a_2)\},\ \{(s_4, a_1), (s_4, a_2)\}\Big\}$. It is easy to see that the resulting partition B given by $B = \Big\{\{(s_1, a_1), (s_1, a_2)\},\ \{(s_2, a_1), (s_3, a_2)\},\ \{(s_2, a_2), (s_3, a_1)\},\ \{(s_4, a_1), (s_4, a_2)\}\Big\}$ is a reward respecting SSP partition. We can derive the quotient MDP $\mathcal{M}|B = \langle S', A', \Psi', P', R' \rangle$ as follows:
$S' = B|S = \Big\{\{s_1\}, \{s_2, s_3\}, \{s_4\}\Big\}$ are the states of $\mathcal{M}/B$.
Now, $\eta(s_1) = 1$, $\eta(s_2) = \eta(s_3) = 2$ and $\eta(s_4) = 1$. Let $A' = \{a'_1, a'_2\}$. Hence we set $A'_{\{s_1\}} = \{a'_1\}$, $A'_{\{s_2, s_3\}} = \{a'_1, a'_2\}$ and $A'_{\{s_4\}} = \{a'_1\}$. Now $P'(\{s_1\}, a'_1, \{s_2, s_3\}) = P(s_1, a_1, s_2) + P(s_1, a_1, s_3) = P(s_1, a_2, s_2) + P(s_1, a_2, s_3) = 1.0$. Proceeding similarly, we have

$$P'(\{s_1\}, a'_1, \{s_2, s_3\}) = 1.0 \qquad P'(\{s_4\}, a'_1, \{s_4\}) = 1.0$$
$$P'(\{s_2, s_3\}, a'_1, \{s_1\}) = 0.8 \qquad P'(\{s_2, s_3\}, a'_2, \{s_1\}) = 0.2$$
$$P'(\{s_2, s_3\}, a'_1, \{s_4\}) = 0.2 \qquad P'(\{s_2, s_3\}, a'_2, \{s_4\}) = 0.8$$

$R'(\{s_2, s_3\}, a'_1) = 0.2$, $R'(\{s_2, s_3\}, a'_2) = 0.8$ and all other rewards are zero. Figure 3.5(b) shows the transition graph for $\mathcal{M}/B$. Note that this MDP is the same as that shown in Figure 3.4(b) except for a relabeling of states and actions. The two MDPs are examples of *isomorphic* MDPs, a notion we will develop further in the next chapter. Now we can define a homomorphism $\langle f, \{g_s | s \in S\}\rangle$ from $\mathcal{M}$ to $\mathcal{M}/B$ as follows: $f(s_1) = \{s_1\}$, $f(s_2) = \{s_2, s_3\}$, $f(s_3) = \{s_2, s_3\}$ and $f(s_4) = \{s_4\}$. $g_{s_1}(a_i) = g_{s_4}(a_i) = a'_1$, for $i = 1, 2$, $g_{s_2}(a_1) = g_{s_3}(a_2) = a'_2$ and $g_{s_2}(a_2) = g_{s_3}(a_1) = a'_1$.

3.6 Relation to Stochastic Bisimulations

This approach to minimization is closely related to earlier work on MDP minimization be Dean and Givan (1997). Their framework is based on the notion of *stochastic bisimulation homogeneity* (Givan et al., 2003). The basic mathematical notion underlying their framework is that of a stochastic bisimulation (Hennessy and Milner, 1985). Formally a stochastic bisimulation is defined as follows:

Definition: Let $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ and $\mathcal{M}' = \langle S', A, \Psi', P', R'\rangle$ be two MDPs with the same action set, with every action being admissible in all the states. Let $E \subseteq S \times S'$ be a relation. We use the notation $E(s, s')$ to indicate that (s, s'), $s \in S$ and $s' \in S'$, belongs to E. E is a *stochastic bisimulation* if each $s \in S$ (and $s' \in S'$) appears in some pair in E, and, whenever $E(s, s')$, both of the following hold for all actions $a \in A$:

1. $R([s]_{E|S}, a)$ and $R'([s']_{E|S'}, a)$ are well defined and equal to each other.

2. For states $t \in S$ and $t' \in S'$ s.t. $E(t, t')$, $T(s, a, [t]_{E|S}) = T'(s', a, [t']_{E|S'})$.

Here $E|S$ is the reflexive, symmetric, transitive closure of E projected onto S. Similarly $E|S'$ is the reflexive, symmetric, transitive closure of E projected onto S'. $R([s]_{E|S}, a)$ is well defined if there exists some $K \in \mathbb{R}$, such that for every $t \in [s]_{E|S}$, $R(t, a) = K$. A stochastic bisimulation from $\mathcal{M}$ to $\mathcal{M}$ is also an equivalence relation on S and leads to a "homogeneous" partition of S. As we shall establish shortly, a homogeneous partition is a reward respecting SSP partition and hence can be used to construct a reduced model of the MDP as described above. Note that the above definition of a stochastic bisimulation considers relations on the state sets. If we extend the definition to relations on $\Psi \times \Psi'$ we can model the same notion of equivalence as that entailed by MDP homomorphisms.

We now formalize the relation between stochastic bisimulations and MDP homomorphisms. For purposes of comparison we look at *state homomorphisms*, i.e., $g_s(a) = a$ for all (s, a) in Ψ. Hence we consider a MDP homomorphism to be a map between state sets in this section. It can be argued that stochastic bisimulations are a more expressive concept than homomorphisms. They allow us to establish a direct correspondence between two MDPs that have some similarity in their transition behavior. But that is not necessarily the case as the following result shows.

Theorem 6: A stochastic bisimulation exists between two MDPs $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ and $\mathcal{M}' = \langle S', A, \Psi', P', R'\rangle$ if and only if there exists an MDP $\mathcal{M}'' = \langle S'', A, \Psi'', P'', R''\rangle$ that is a (state) homomorphic image of both $\mathcal{M}$ and $\mathcal{M}'$.

Proof: ($\Rightarrow$) Let $E \subseteq S \times S'$ be a stochastic bisimulation between $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ and $\mathcal{M}' = \langle S', A, \Psi', P', R'\rangle$. We show by construction that there exists $\mathcal{M}'' = \langle S'', A, \Psi'', P'', R''\rangle$ which is a homomorphic image of both $\mathcal{M}$ and $\mathcal{M}'$.

First, we establish that $E|S$ is a reward respecting SSP partition of $\mathcal{M}$. By definition of a stochastic bisimulation we have that $R([s]_{E|S}, a)$ is well-defined for all $a \in A$. Hence $E|S$ is reward respecting. Let $s_1, s_2 \in S$ be s.t. $[s_1]_{E|S} = [s_2]_{E|S}$. This implies that there exists a "path" in E from s_1 to s_2, ignoring arc directions. Without loss of generality let us assume that for every $s' \in S'$, if $E(s_1, s')$ then $E(s_2, s')$.[3] For every $t \in S$ and $t' \in S'$ s.t. $E(t, t')$, the second condition of stochastic bisimulation states that $T(s_1, a, [t]_{E|S}) = T'(s', a, [t']_{E|S'}) = T(s_2, a, [t]_{E|S})$ for all $a \in A$. Thus we have $T(s_1, a, [t]_{E|S}) = T(s_2, a, [t]_{E|S})$ for all $[t]_{E|S} \in E|S$ and for all $a \in A$. Thus $E|S$ is an reward respecting SSP partition.

Let $\mathcal{M}'' = \langle S'', A, \Psi'', P'', R'' \rangle$ be the quotient MDP $\mathcal{M}/E|S$. Let $f : S' \to E|S$, be given by $f(s') = [s_1]_{E|S}$ if $E(s_1, s')$. The map f is a surjection, since for every $s' \in S'$, there exists at least one $s_1 \in S$ s.t. $E(s_1, s')$. It is well defined: if there exist $s_1, s_2 \in S$, s.t. $E(s_1, s')$ and $E(s_2, s')$, then $[s_1]_{E|S} = [s_2]_{E|S}$. From the second condition of a stochastic bisimulation, for every $t \in S$ and $t' \in S'$ s.t. $E(t, t')$ we have $T'(s', a, [t']_{E|S'}) = T(s_1, a, [t]_{E|S}) = P''([s_1]_{E|S}, a, [t]_{E|S}) = P''(f(s'), a, f(t'))$ for all $a \in A$. From the first condition we have that $R'(s', a) = R(s_1, a) = R''([s_1]_{E|S}, a) = R''(f(s'), a)$ for all $a \in A$. Thus f is a homomorphism from $\mathcal{M}'$ to $\mathcal{M}/E|S$.

Therefore $\mathcal{M}/E|S$ (similarly $\mathcal{M}'/E|S'$) is a homomorphic image of both $\mathcal{M}$ and $\mathcal{M}'$.

($\Leftarrow$) Let $\mathcal{M}''$ be a homomorphic image of $\mathcal{M}$ under f and a homomorphic image of $\mathcal{M}'$ under f'. Define a relation $E \subseteq S \times S'$ s.t. $E(s, s')$ if and only if $f(s) = f'(s')$. Note that $E|S$ is B_f and $E|S'$ is $B_{f'}$. Therefore $R([s]_{E|S}, a)$ and $R'([s']_{E|S'}, a)$ are well-defined. Also if $E(s, s')$ then $R([s]_{E|S}, a) = R''(f(s), a) = R''(f'(s'), a) = R'([s']_{E|S'}, a)$ for all $a \in A$. For every $t \in S$ and $t' \in S'$ s.t. $E(t, t')$ we have $T(s, a, [t]_{E|S}) = T''(f(s), a, f(t)) = T''(f'(s'), a, f'(t')) = T'(s', a, [t']_{E|S'})$. Therefore E is a stochastic bisimulation between $\mathcal{M}$ and $\mathcal{M}'$. □

[3]Strictly speaking we should consider induction on the length of the path in E in the proof. The resulting properties are the same under the above assumption as under induction. In the interests of clarity and brevity we make this assumption.

Corollary: Let $f : S \rightarrow S'$ be a state homomorphism from $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to $\mathcal{M}' = \langle S', A, \Psi', P', R' \rangle$. The relation $E \subseteq S \times S'$, defined by $E(s, s')$ if and only if $f(s) = s'$, is a stochastic bisimulation.

Proof: Given that f is a state homomorphism from $\mathcal{M}$ to $\mathcal{M}'$. Every MDP is homomorphic to itself under the identity map. Thus $\mathcal{M}$ and $\mathcal{M}'$ share a homomorphic image, namely $\mathcal{M}'$. From the construction in the proof of Theorem 6, we have that a stochastic bisimulation, E, between $\mathcal{M}$ and $\mathcal{M}'$ is defined by the relation $E(s, s')$ if an only if $f(s) = f'(s')$, which in this case reduces to $f(s) = s'$. □

The above results establish that every MDP state homomorphism induces a stochastic bisimulation and every stochastic bisimulation can be modeled by a pair of MDP state homomorphisms. It is straightforward to establish corresponding results for MDP homomorphisms and stochastic bisimulations on the admissible state-action pairs. Hence for the purposes of minimization, bisimulations do not provide any additional power. But we believe MDP homomorphisms are a simpler notion than bisimulations and help us to better understand the minimization process. Employing homomorphisms allows us to easily cast various abstraction schemes as special cases of MDP minimization.

3.7 Learning With Reduced Models

Traditionally minimization methods are thought of as a suitable pre-processing step to planning methods, since both need complete specification of the model. As has been demonstrated in the past, even in the presence of a complete model it is advantageous to employ real-time dynamic programming or reinforcement leaning to solve especially large MDPs, since these methods focus the search for the solution on a relatively small but relevant area of the state space. Employing minimization as a pre-processing step in such a scenario saves us further effort since we are now dealing with a (possibly) smaller model.

In the Section 4.2 we show that under certain conditions we can derive reduced models without having to enumerate the entire state space. Even if we do not know the complete system model apriori, we often have sufficient prior knowledge on structural redundancy to derive abstract representation of MDPs using minimization ideas. In particular in Chapter 6 we explore an approach where we can derive the reduced model with limited experience in the "real-world" and then employ this model in learning policies in the original MDP. Also casting abstraction as a minimization problem helps us in achieving greater insight into many existing abstraction approaches and in developing new ones, even if the approach itself does not conform exactly to the parameters of a minimization technique. In other words, we obtain useful abstract representations for the problem, but do not derive minimal models, and do not require complete knowledge of the system model.

In this work we adopt the notion of homomorphic equivalence of state-action pairs to derive reduced models. Other notions of equivalence have been adopted in the literature,[4] two of which are useful and in some sense more powerful than homomorphic equivalence. The first is optimal value equivalence, where two state action pairs are considered equivalent if their optimal action values are the same. In other words, $(s_1, a_1) \equiv_{Q^\star} (s_2, a_2)$ if and only in $Q^\star(s_1, a_1) = Q^\star(s_2, a_2)$. It can be shown that the coarsest reward respecting SSP partition is a refinement of $B_{Q^\star}$. Thus optimal value equivalence might lead to a smaller quotient model than homomorphic equivalence. The chief drawback is that it is not possible to determine $B_{Q^\star}$ from the problem parameters directly. Most often we need to solve the problem before we can determine $B_{Q^\star}$. Still it is a useful notion and some abstraction approaches try to estimate $B_{Q^\star}$ incrementally. They start with some coarse partition and successively

[4]Givan et al. (2003) discuss several such notions in some detail and put forth arguments as to why stochastic bisimulation is a better notion of equivalence. Those arguments apply here too, since, as demonstrated earlier, homomorphic equivalence is the same as stochastic bisimulation equivalence.

refine the partition with more learning or planning (e.g., Whitehead and Ballard, 1991; McCallum, 1995; Jonsson and Barto, 2001; Kim and Dean, 2001; Feng et al., 2003).

The other notion of equivalence is optimal policy equivalence. Here two state action pairs are considered equivalent if the probability of picking those actions in those states under some optimal policy are same. In other words, $(s_1, a_1) \equiv_{\pi^\star} (s_2, a_2)$ if and only if $\pi^\star(s_1, a_1) = \pi^\star(s_2, a_2)$ for some $\pi^\star$. Both the coarsest reward respecting SSP partition and $B_{Q^\star}$ are refinements of $B_{\pi^\star}$. As with $B_{Q^\star}$, using $B_{\pi^\star}$ requires that we use some form of incremental estimation.

Both of these notions of equivalence are particularly attractive if we have little prior knowledge about the structure of the problem being solved. We any way estimate the optimal policy and if we are using a value based solution approach, we estimate the optimal value function. So why not use them as the basis for abstraction also? But it is seldom the case that we start with absolutely no knowledge of the problem. We often have some information about inherent symmetry in the problem and we can also make certain independence assumptions about the various features describing the problem. With such structural knowledge it is easier to specify reduced images that exploit homomorphic equivalence than it is for other forms of equivalence. We shall see examples of this in the later sections. However note that homomorphic equivalence does imply both optimal value and optimal policy equivalence. We can also combine different notions of equivalence and start with a reward respecting SSP partition and coarsen it incrementally to achieve a more compact model corresponding to $B_{Q^\star}$ or $B_{\pi^\star}$.

3.8 Related Work

There has been extensive work in algebraic minimization of finite state machines. Hartmanis and Stearns (1966) present a excellent introduction to the notions of machine homomorphisms, SP partitions, FSA minimization and decomposition techniques. The basic FSA minimization approach consists of identifying the coarsest partition of the state set that satisfies the substitution property. This is the analogue of the SSP property for deterministic FSA. Hartmanis and Stearns outline a method to identifying the coarsest SP partitions based on the partial order on all SP partitions. The method requires us to specify some SP partition of the system before hand. Minimization algorithms for Markov chains follow similar lines, with the equivalence criterion of *lumpability*, which is related to the substitution property. Kemeny and Snell (1960) discusses equivalence of Markov chains in detail. They also describe weak lumpability, a relaxation of the equivalence criterion.

Minimization approaches for other modeling paradigms, such as probabilistic automata (Paz, 1971) and probabilistic transition systems (Larsen and Skou, 1991), are usually based on the notion of bisimulation and its stochastic extension. Bisimulations were introduced by Hennessy and Milner (1985) and are a many to many map between the state sets of two structures. They help to establish equivalence among the elements, such that some aspect of the transition structure of the systems are preserved when equivalent states are aggregated.

Checking models of concurrent processes, popularly called *model checking*, is another field that widely employ minimization ideas (McMillan, 1993; Lee and Yannakakis, 1992; Ip and Dill, 1996; Emerson and Sistla, 1996; Emerson and Trefler, 1998). Researchers employ various logical models of concurrent programs and systems and check them for correctness, i. e. if the program or system really does what it is supposed to do. For example, if we are designing a system with an abort switch, we want some assurance that when we throw the switch, the system really does abort and does so in a "clean" way. The basic model checking process is not very relevant to our work, but model checking systems frequently employ some form of minimization. The goal is to derive a smaller model of the system, whose correctness implies the correctness of the original model. Lee and Yannakakis (1992) base their approach to minimization on bisimulations and Dean and Givan (1997) extend it to MDP minimization.

Dean, Givan and colleagues have explored MDP minimization in detail. Their model minimization framework (Dean and Givan, 1997) is based on the notion of *bisimulation homogeneity*, which is equivalent to the SSP property restricted to the state set of the MDP. They develop a simple algorithm that successively refines violations of homogeneity and produces the coarsest homogeneous partition of the state set. The quotient MDP may then be constructed along similar lines as in Section 3.4. They establish many theoretical results (Givan et al., 2003) on the equivalence of value functions and on the correctness of their algorithm. Many of the results we presented in this chapter are extensions of their results to our framework.

Dean and Givan also examine several MDP abstraction algorithms (Dean and Givan, 1997; Givan et al., 2003; Givan and Dean, 1997) and show them to be instances of their minimization approach applied to special representation schemes. The various algorithms take advantage of the structure in the system to develop polynomial time algorithms for minimization. Boutilier and Dearden's (1994) state aggregation method employs Boolean features to represent the state space and look for homogeneous partitions among those described by logic formulae on the features. This is equivalent to searching for a homomorphism among projections onto a subset of features. Structured policy iteration of Boutilier et al. (1995) is a policy iteration algorithm that implicitly does state abstraction. They represent the MDP via DBNs and the CPTs and value function as decision trees. Dean and Givan show that their algorithm implicitly computes homogeneous partitions among those partitions representable by decision trees on the state features. In a recent presentation Givan (Parr and Givan, 2001) mentions that Boutilier et al.'s (2001) symbolic dynamic programming (SDP) is also an instance of an implicit minimization algorithm. SDP employs situational calculus for state representations and searches for homogeneous partitions among partitions representable as first order logic formulae in the calculus.

These approaches do not employ state-action equivalence, but Dean et al. (1998) do consider homogeneous partitions of the state-action space in their minimization algorithm. But they employ the traditional definition of projection of partitions. As Example 1 in Chapter 2 demonstrated, this is a weaker concept and hence does not lead to the greater reductions facilitated by our approach.

CHAPTER 4
SYMMETRY, STRUCTURE AND APPROXIMATIONS

In this chapter we further explore the power of MDP homomorphisms. We develop an inclusive definition of symmetries in MDPs and show that this results in a special case of homomorphic equivalence. We then explore several special forms of homomorphisms suited for structured MDPs, where we exploit independence between features describing the state set. In many cases, even when the homomorphism conditions are not met exactly, we can form useful abstractions using some relaxed notion of equivalence. We develop two forms of approximate homomorphisms that allow us to bound the loss when forming such abstractions.

4.1 Modeling Symmetries

Researchers in AI have long recognized the usefulness of abstracting away symmetry (Amarel, 1968) in a problem description. Informally, a symmetric system is one which is invariant under certain transformations onto itself. An obvious class of symmetries is based on geometric transformations, such as reflections, rotations and translations. An example of a reflectional symmetry in an MDP settings was shown in the example in Figure 1.2. But invariance often arise due to many other properties of a system, especially structural properties. One of the interesting class of such symmetries that we will revisit later in this thesis is that due to object interchangeability. Informally, these classes of symmetry arise when you can replace some objects in the world with other similar objects and the dynamics of the world does not change as far as achieving your primary objective is concerned. Such systems are usually not thought of as being symmetric systems, but our definition of symmetry treats object interchangeability the same way as it would reflection or rotation.

In this section we formalize the notion of MDP symmetries employing group theoretic concepts. Since we appeal to only the underlying mathematical structure of the problem, this is a very inclusive definition of symmetry that is applicable to a variety of problem domains. We also show that symmetric equivalence can be accommodated naturally in our minimization framework and is in fact a special case of homomorphic equivalence.

4.1.1 Symmetry Groups

Symmetries of a structure are usually characterized by the *symmetry group* — the group of all automorphisms of the structure. Automorphisms are transformations of a structure onto itself such that all the properties of the structure are preserved. We first define MDP automorphisms and then symmetry groups of MDPs.

Definition: An MDP homomorphism $h = \langle f, \{g_s | s \in S\}\rangle$ from MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ is an *MDP isomorphism* from $\mathcal{M}$ to $\mathcal{M}'$ if and only if f and g_s, $s \in S$, are bijective. $\mathcal{M}$ is said to be *isomorphic* to $\mathcal{M}'$ and vice versa.

Note that property (1) of a homomorphism reduces to a simpler form in this case: $P(s, a, s') = P'(f(s), g_s(a), f(s'))$ for all $s, s' \in S$ and $a \in A_s$. Therefore, when two MDPs are isomorphic, it means that the MDPs are the same except for a relabeling of the states and a state-specific relabeling of the actions. Thus we can transfer policies learned for one MDP to the other by simple transformations. Also note that an MDP $\mathcal{M}$ is a minimal MDP if all $\mathcal{M}'$ that are homomorphic to $\mathcal{M}$ are also isomorphic to it.

Definition: An MDP isomorphism from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to itself is an *automorphism* of $\mathcal{M}$.

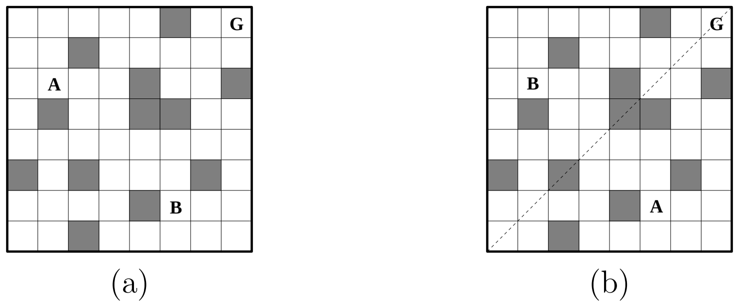


Figure 4.1. (a) A symmetric gridworld problem. Reproduced from Figure 1.2. (b) Reflection of the gridworld in (a) about the *NE-SW* diagonal.

Intuitively one can see that automorphisms can be used to describe symmetries in a problem specification. In the example of Figure 1.2(a), a reflection of the states about the NE-SW diagonal and a swapping of actions N and E and of actions S and W is an automorphism. It is easy to see that this mapping captures the symmetry discussed earlier. Figure 4.1 shows both the original and the reflected MDP.

Proposition: The set of all automorphisms of an MDP $\mathcal{M}$, denoted by Aut$\mathcal{M}$, forms a group under composition of homomorphisms. This group is the *symmetry group* of $\mathcal{M}$.

Let $\mathcal{G}$ be a subgroup of Aut$\mathcal{M}$ denoted by $\mathcal{G} \leq \text{Aut}\mathcal{M}$. The subgroup $\mathcal{G}$ defines an equivalence relation $\equiv_{\mathcal{G}}$ on Ψ: $(s_1, a_1) \equiv_{\mathcal{G}} (s_2, a_2)$ if and only if there exists $h \in \mathcal{G}$ such that $h(s_1, a_1) = (s_2, a_2)$. Note that since $\mathcal{G}$ is a subgroup, this implies that there exists a $h^{-1} \in \mathcal{G}$ such that $h^{-1}(s_2, a_2) = (s_1, a_1)$. Let $B_{\mathcal{G}}$ be the partition of Ψ induced by $\equiv_{\mathcal{G}}$. We need the following lemma to prove Theorem 7:

Lemma: For any $h = \langle f, \{g_s | s \in S\}\rangle \in \mathcal{G}$, $f(s) \in [s]_{B_{\mathcal{G}}|S}$.

Proof: The lemma follows from the properties of groups (Lang, 1967), namely closure and existence of an inverse. $\square$

Theorem 7: Let $\mathcal{G} \leq \text{Aut}\mathcal{M}$ be a subgroup of automorphisms of $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$. The partition $B_{\mathcal{G}}$ is a reward respecting SSP partition of $\mathcal{M}$.

Proof: Consider (s_1, a_1), $(s_2, a_2) \in \Psi$ such that $(s_1, a_1) \equiv_{\mathcal{G}} (s_2, a_2)$. This implies that there exists an $h = \langle f, \{g_s | s \in S\} \rangle$ in $\mathcal{G}$ such that $f(s_1) = s_2$ and $g_{s_1}(a_1) = a_2$.

From the definition of an automorphism we have that for any $s \in S$, $P(s_1, a_1, s)$ $= P(s_2, a_2, f(s))$. Using the lemma, $\sum_{s' \in [s]_{B_{\mathcal{G}}|S}} P(s_1, a_1, s') = \sum_{s' \in [s]_{B_{\mathcal{G}}|S}} P(s_2, a_2, s')$. Since we chose s arbitrarily, this holds for all s in S. Hence $B_{\mathcal{G}}$ is an SSP partition. Again from the definition of an automorphism we have that $R(s_1, a_1) = R(s_2, a_2)$. Hence $B_{\mathcal{G}}$ is reward respecting too. □

Corollary 1: Let $\mathcal{G} \leq \text{Aut}\mathcal{M}$ be a group of automorphisms of $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$. There exists a homomorphism $h^{\mathcal{G}}$ from $\mathcal{M}$ to some $\mathcal{M}'$, such that the equivalence relation induced by $h^{\mathcal{G}}$, $\equiv_{h^{\mathcal{G}}}$, is the same relation as $\equiv_{\mathcal{G}}$.

Proof: We can prove this by constructing a homomorphism $h^{\mathcal{G}}$ from $\mathcal{M}$ to $\mathcal{M}|B_{\mathcal{G}}$, given by $h^{\mathcal{G}} = \langle f, \{g_s | s \in S\} \rangle$ where $f(s) = [s]_{B_{\mathcal{G}}|S}$ and $g_s(a) = a'_i$ such that $T(s, a, [s']_{B_{\mathcal{G}}|S}) = P'([s]_{B_{\mathcal{G}}|S}, a'_i, [s']_{B_{\mathcal{G}}|S})$ for all $[s']_{B_{\mathcal{G}}|S} \in B_{\mathcal{G}}|S$. In other words, if $[(s, a)]_{B_{\mathcal{G}}|S}$ is the i-th unique block in the ordering used in the construction of $\mathcal{M}/B_{\mathcal{G}}$, then $g_s(a) = a'_i$. It is easy to verify that $B_{h^{\mathcal{G}}} = B_{\mathcal{G}}$. □

The image of $\mathcal{M}$ under $h^{\mathcal{G}}$ is called the *$\mathcal{G}$-reduced image* of $\mathcal{M}$. We say state action pairs (s_1, a_1) and $(s_2, a_2) \in \Psi$ are *symmetrically equivalent* if for some $\mathcal{G} \leq \text{Aut}\mathcal{M}$, $(s_1, a_1) \equiv_{\mathcal{G}} (s_2, a_2)$.

Corollary 2: For any symmetrically equivalent $(s_1, a_1), (s_2, a_2) \in \Psi$, $Q^\star(s_1, a_1) = Q^\star(s_2, a_2)$ and hence the optimal action-value function of a symmetric MDP is also symmetric, i. e., invariant under the transformations in the symmetry group of $\mathcal{M}$.

Corollary 3: If $\pi'^\star$ is an optimal policy for some $\mathcal{G}$-reduced image of MDP $\mathcal{M}$, then $\pi'^\star_{\mathcal{M}}$ is an optimal policy for $\mathcal{M}$.

Note that the converse of Theorem 7 is not true. It is possible to define SSP partitions that are not generated by groups of automorphisms. Frequently the Aut$\mathcal{M}$-reduced model of an MDP is a minimal image. We look to taking advantage of structure inherent in a symmetry group and the related equivalence classes in deriving symmetrically reduced images. This is a theme we will return to often in this work.

Illustration of Minimization: A Symmetric Abstract MDP Example

Let us return to the MDP $\mathcal{M}$ in Figure 3.5(a). The reduced MDP $\mathcal{M}/B$ shown in Figure 3.5(b) is also the the Aut$\mathcal{M}$-reduced image of $\mathcal{M}$. Let $\mathcal{I}$ be the identity map on Ψ and let h be the automorphism on $\mathcal{M}$ defined by: $h(s_1, a_1) = (s_1, a_2)$, $h(s_2, a_1) = (s_3, a_2)$, $h(s_2, a_2) = (s_3, a_1)$ and $h(s_4, a_1) = (s_4, a_2)$. The symmetry group of $\mathcal{M}$, Aut$\mathcal{M}$, is $\{\mathcal{I}, h\}$ with the composition operator. The partition induced by the symmetry group is $B_{\text{Aut}\mathcal{M}} = \Big\{\{(s_1, a_1), (s_1, a_2)\},\ \{(s_2, a_1),\ (s_3, a_2)\},$ $\{(s_2, a_2), (s_3, a_1)\},\ \{(s_4, a_1), (s_4, a_2)\}\Big\}$, which is the same as B from the previous example.

Example of Reductions Not Modeled By Symmetry Groups

Figure 4.2(a) shows a very simple abstract MDP with a non-trivial symmetry group. Each of the states depicted have just one action with the dynamics as shown in the figure. The action causes a transition from state S to one of states A, B or C with equal probability. From each of the states, the action transitions to the absorbing state G with probability 1 and obtains a reward of $+1$. The symmetry group for this MDP consists of all permutations of the states A, B and C. The coarsest reward respecting partition for this MDP is: $\Big\{\{S\}, \{A, B, C\}, \{G\}\Big\}$. Since there is only one action, we have not indicated that here. The minimal image of this MDP is shown in Figure 4.2(c).

Figure 4.2(b) shows a similar MDP, with slightly different dynamics. Here the action from state S causes transition to states A, B and C with different probabilities.

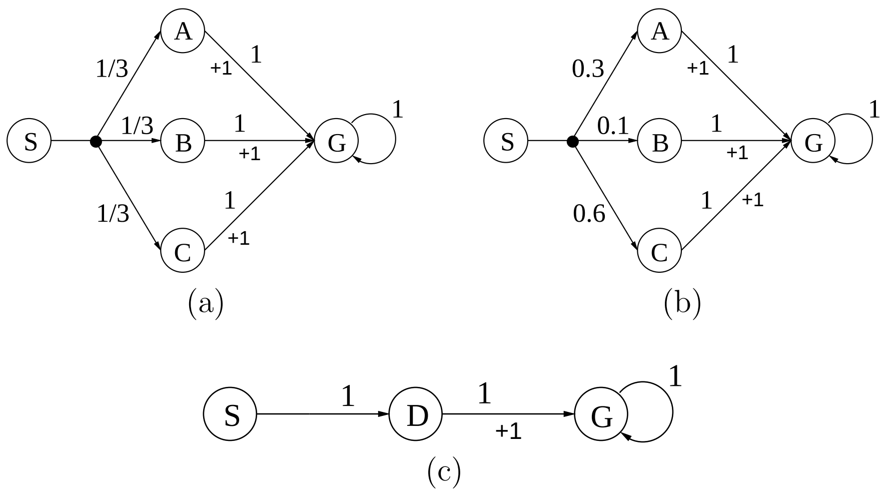


Figure 4.2. (a) Transition graph of a symmetric MDP. (b) Transition graph of a similar MDP, but with a trivial symmetry group. (c) Minimal image for both the MDPs.

Therefore this MDP has only a trivial symmetry group consisting of just the identity map. But, as with the other MDP, the coarsest reward respecting partition for this MDP is: $\Big\{\{S\}, \{A, B, C\}, \{G\}\Big\}$. The MDP in Figure 4.2(c) is a minimal image of this MDP also.

This example demonstrates that not all reductions are generated by symmetry groups. Another point to note in this example is that the MDP which has a non-trivial symmetry group has a repeated structure. When we aggregate states together while constructing a reduced model, the homomorphism conditions require only that for each action in a state there is some other action from an equivalent state which has the same block transition behavior. When the reductions arise from symmetry groups for each action in a state there is some other action from an equivalent state which has the same transition behavior with respect to each member of a given block. Therefore not only are the homomorphism conditions satisfied, but a stronger condition is met. In practice though symmetric reductions arise often and can be identified by a cursory examination of system properties unlike non-symmetric reductions. All the examples we encounter in the later chapters employ symmetric reductions.

4.1.2 Discussion on Identifying Symmetry Groups

In this section we have established that abstracting away symmetry in a problem is equivalent to finding a homomorphic image of the MDP. This implies that we can treat symmetries as a special case of homomorphic reductions and do not need special mechanisms to handle them. In reality we can take advantage of the repeated structure in the problem that a symmetry group captures to derive more efficient algorithms. It is interesting to note that most of the reductions we are interested in, as we shall see in later chapters, arise from symmetries of a system, so much so that people tend to confuse homomorphisms and symmetries.

In the previous section we outlined a polynomial time algorithm to identify reward respecting SSP partitions. While that algorithm also detects partitions induced by symmetry groups, it is not an easy task to specialize the algorithm to look for symmetry groups of MDPs. We need to make certain structural assumptions about the MDP to derive tractable algorithms for symmetry group identification. In the next section we explore factored MDPs and introduce special forms of automorphisms and homomorphisms. By restricting our search to such special forms, it is possible to derive more efficient abstraction algorithms.

In a large family of tasks the symmetry groups are known beforehand or can be specified by the designer through a superficial examination of the problem. In Section 4.2 we present an algorithm that efficiently constructs a reduced MDP given a symmetry group, without having to enumerate all the states and actions explicitly. In learning problems, it is possible achieve some speedup in the process even if we have incomplete knowledge of homomorphisms and symmetry groups of the problem. We explore this direction in greater detail in Chapters 5 and 6.

4.1.3 Related Work

The MDP minimization algorithms analyzed by Givan et al. (2003) do not consider symmetries of MDPs. While it is possible to extend these algorithms to accommodate symmetric equivalence of states, without considering state-action equivalence they cannot model many interesting kinds of symmetry. Symmetry groups have been employed in the minimization of other modeling paradigms. Jump (1969) employs symmetry groups in decomposition of FSAs. His approach is an extension of an approach proposed by Hartmanis and Stearns (1966) based on a notion related to SP partitions. Glover (1991) employs symmetry groups for deriving *shift invariant* models of Markov chains. The goal here is to form a representation of a Markov chain that behaves similarly under sequences of inputs that are similar but shifted in time by varying amounts.

Model checking literature abounds with examples of exploiting symmetry in minimization. Most models of concurrent systems employ a factored representation, with a feature for each process that indicates the current state of the process. The execution of the system is graphically modeled as a *Kripke* structure with the nodes of the structure representing states of the system and the edges possible deterministic transitions (McMillan, 1993). Minimization algorithms for Kripke structures focus on the special class of automorphisms that can be expressed as a permutation of the feature values. Thus the symmetry group of concurrent process model will be given by permutations of the feature values that leave the Kripke structure unaltered (Emerson and Sistla, 1996; Ip and Dill, 1996; Emerson and Sistla, 1997; Emerson and Trefler, 1998; Emerson et al., 1997). Of particular interest to us is the work by Emerson and Sistla (1996) in which they present an incremental algorithm for building a quotient structure. In Table 4.2 we extend their algorithm to MDPs, by incorporating rewards and stochasticity and are looking to further extend it to operate with a sample model. Emerson and Trefler (1999) develop many relaxed symmetry criteria which lead to useful reductions in the problem size. We look to extending their results to factored MDPs and achieving similar reductions in problem size.

Researchers in AI have long recognized the usefulness of abstracting away symmetry. The body of relevant literature on approximation is huge and it is fairly impossible to present a complete survey here. We just mention a couple of works, one old and one recent to indicate the continuing interest in this area. Amarel (1968) discusses the Missionaries and Cannibals problem in detail, exploring various representation schemes that lead to increasingly more efficient solutions. One of the schemes abstracts away the *time reversal* symmetry in the problem. It does not matter if you are going from the left bank to right bank or vice versa. So we can partly solve the problem of going from one bank to the other and derive the complete solution by using the time reversed partial solution for going in the other direction. This leads to significant speed up in the planning procedure. Popplestone and Grupen (2000) take a system theoretic approach to modeling permutation symmetries in the same problem domain, to abstract away the identity of the various people in the problem. They model the system dynamics by *generalized transfer functions* (GTF) and examine how various symmetries in the problem domain can be modeled as symmetries of the GTF. They look at classes of symmetries generated by permuting the inputs and the outputs separately and then permuting both simultaneously. They employ the symmetry in generating a quotient structure that ignores the identity of the persons involved and works only with relative numbers.

While various abstraction techniques have been successfully employed with RL algorithms, there is not much work on explicitly employing symmetries. Zinkevich and Balch (2001) define symmetries of MDPs employing equivalence relations on the state-action pairs, but they do not make connections to group theoretic concepts or to minimization algorithms. They show that the optimal action-value function of a symmetric system is symmetric and suggest that the symmetrically equal action-value function entries be duplicated. They also study in some detail symmetries that arise in multi-agent systems. They restrict their analysis to flat RL systems and do not consider hierarchical systems. Drummond (1998) employs a visual processing technique to detect useful subgoals from the value function that a RL agent learns and defines suitable macro-actions to achieve these subgoals. He employs various transformations to the subgoals detected to identify similar situations and can detect symmetrically equivalent situations also. But his method is limited to 2-D environments presently and ones in which the symmetry is visually apparent in the value function. Hengst (2002) presents AHRL, a hierarchical decomposition algorithm for factored MDPs. AHRL can exploit symmetry in the environment to define subproblems, where symmetry is identified by "repeatability". His approach presently is severely limited in applicability and works only with special representation schemes.

4.2 Homomorphisms of Factored MDPs

The framework we have developed thus far assumes a monolithic representation of an MDP. But many classes of problems that are modeled as MDPs often have some inherent structure. We can exploit this structure using a feature based or factored representation, and there are many efficient MDP solution techniques that take advantage of such representations (e.g. Boutilier et al., 1999). Similarly we can model abstractions that arise from structured homomorphisms and exploit the structure to derive more efficient minimization algorithms and compact representations of reduced models and symmetry groups.

Another reason that drives us to look at special classes of homomorphisms is that the polynomial time complexity of the algorithm presented in the previous chapter results from assuming that determining membership in a block of a partition takes constant time. This is not generally true and in fact, depending on how the MDP is represented, this might be NP-hard. Dean and Givan (1997) clearly demonstrate this difficulty with certain representation schemes.

Let us consider one such scheme used by a state-aggregation method (Boutilier and Dearden, 1994). The states of the MDP are represented by several boolean features. Each state in the MDP is represented by a specific assignment to these boolean features. The partitions then are represented as boolean formulae. Determining membership in these partitions, to check if they are empty or not, is boolean formulae satisfiability. This is known to be NP-complete and hence deriving the minimal image of an MDP in such cases is NP-complete. Dean and Givan examine algorithms that obtain reasonable reductions in polynomial time, in special cases which use structured representations for the MDP parameters.

4.2.1 Structured MDPs

A structured MDP is described by the tuple $\langle S, A, \Psi, P, R\rangle$. The state set S is now given by M features or variables, $S \subseteq \prod_{i=1}^{M} S_i$, where S_i is the set of permissible values for feature i. Thus any $s \in S$ is of the form $s = \langle s_1, \ldots, s_M\rangle$, where $s_i \in S_i$ for all i.[1] A state s can also be thought of as an unique assignment to the state variables s_i.

The transition probability matrix P is usually described by two-slice *dynamic Bayesian networks* (2-DBNs) (Dean and Kanazawa, 1989). A 2-DBN is a two layer directed acyclic graph, one for each action, whose nodes are $\{s_1, \ldots, s_M\}$ and $\{s'_1, \ldots, s'_M\}$. Here s_i denotes the value of feature i at the present state and s'_i denotes the value of feature i in the resulting state. Many classes of structured problems, as in the example below, may be modeled by a DBN in which the arcs are restricted to go

[1]The action set may also be similarly structured, defined via K components, $A \subseteq \prod_{i=1}^{K} A_i$, where A_i is the set of permissible values for action component i. Presently, we only consider monolithic actions.

from nodes in the first set to those in the second. The state-transition probabilities can be factored as:

$$P(s, a, s') = \prod_{i=1}^{M} \text{Prob}(s'_i|\text{Parents}(s'_i, a))$$

where Parents(s'_i, a) denotes the parents of node s'_i in the DBN corresponding to action a and each of the Prob(s'_i|Parents(s'_i, a)) is given by a conditional probability table (CPT) associated with node s'_i. This is the standard representation of transition probabilities in factored MDPs. In computing the conditional probabilities it is implicitly assumed that the nodes in Parents(s'_i,a) are assigned values according to s. In cases we want to make the dependence of the probability on the previous state explicit, we write Prob(s'_i|Parents$_s$(s'_i, a)).

The reward function too may be similarly represented. Another representation for the reward is to assume that the reward is the sum of linear functions on subset of the features. Thus $R(s, a) = \sum_{i=1}^{L} r_i(s, a)$ where r_i is linear and depends only on a subset of features (Koller and Parr, 2000).

Example

Let us consider a toy robot domain to illustrate our ideas in this section. The task of the robot is to deliver coffee to an office on a rainy day. The state of the robot is described by the following boolean features: HC - the robot has coffee, Loc - true when the robot is in the office, false if it is in the cafe, HU - the robot has an umbrella. The robot has the following actions available: *go* - toggles the location of the robot with probability 0.9, *dc* - deliver coffee, results in a reward of +1 if $HC \wedge Loc$ is true and sets HC to false with probability 0.8, *gu* - get umbrella, sets HU to true, if Loc is true, with probability 0.75, *nop* - do nothing. In cases the actions fail, they leave the state unaltered. This is a modification of the problem described in Boutilier et al.

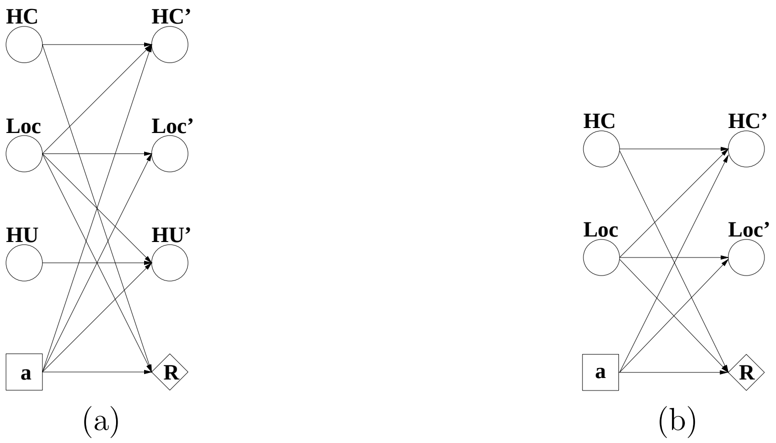


Figure 4.3. (a) The DBN of the coffee robot domain described in the text. The a node is the action node and can take values *dc, go, gu, nop.* (b) The DBN of the homomorphic image generated by a simple projection on to a subset of features.

(1995). The transition dynamics of the system is given by the DBN shown in Figure 4.3(a). □

4.2.2 Structured Morphisms

Adding structure to the state space representation allows us to consider morphisms that are structured, i.e., surjections from one structured set to another. An example of a structured morphism is a simple projection onto a subset of features. We introduce some notation, after Zeigler (1972), to make the following definitions easier. Given a structured set $X \subseteq \prod_{i=1}^{M} X_i$, the *i-th projection* of X is a mapping $\rho_i : X \rightarrow X_i$, defined by $\rho_i(\langle x_1, \ldots, x_M \rangle) = x_i$. We extend this definition to that of a projection on a subset of features. Given a set $J \subseteq \{1, \ldots, M\}$ the *J-projection* of X is a mapping $\rho_J : X \rightarrow \prod_{j \in J} X_j$, defined by $\rho_J = \prod_{j \in J} \rho_j$.

Definition: A *simple projection homomorphism* h from a structured MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to a structured MDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$, with $h(s, a) = (f(s), g_s(a))$, where $f = \rho_F : S \rightarrow S'$, where $F \subseteq \{1, \ldots, M\}$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that, $\forall s, s' \in S, a \in A_s$:

$$\begin{aligned} P'(f(s), g_s(a), f(s')) &= T(s, a, [s']_{B_h|S}) && (4.1) \\ &= \prod_{j \in F} \text{Prob}(s'_j | \text{Parents}(s'_j, a)) && (4.2) \\ R'(f(s), g_s(a)) &= R(s, a). && (4.3) \end{aligned}$$

The first condition implies that F must be such that for all $j \in F$ and $(s, a) \in \Psi$, $s_i \in \text{Parents}(s'_j, a)$ implies $i \in F$. In other words, the DBN of $\mathcal{M}'$ is a sub-graph of the DBN of $\mathcal{M}$, such that no *incoming* arc to the nodes in the sub-graph is cut. The second condition requires that no incoming arc to the reward node is cut either.

Example Continued

Going back to our toy robot example, a projection onto the *HC* and *Loc* features with $g_s(a) = a$ for all $s \in S$ and $a \neq gu$ and $g_s(gu) = nop$ for all $s \in S$ is a projection homomorphism. The DBN of the image MDP is shown in Figure 4.3(b).

An algorithm to determine simple projection homomorphisms of factored MDPs that runs in time polynomial in the number of features is shown in Table 4.1. This algorithm assumes that the MDP dynamics is specified as a 2-DBN. The intuition behind the algorithm is similar to the minimization algorithm from Chapter 3. We start with the reward node in the image MDP. Then we iteratively add nodes and connections that directly influence the nodes in the image MDP. This is equivalent to starting with the partition induced by the reward function and successively refining it, till we achieve a reward respecting SSP partition of the state space. The difference in this case is that we end up the coarsest simple projection image and not the minimal image of the MDP.

It is evident that the space of simple projections is much smaller than that of general maps and may not contain a homomorphism reducing a given MDP. A more

1. Start the queue of nodes to be examined with the reward nodes of the DBN.
2. For node n in the queue, add to the set of nodes in the image DBN any node s_i such that there exists an arc from s_i to n. Add s'_i to the set of nodes in the image DBN and to the queue. Add the arc from s_i to n and s_i to s'_i (if applicable) to the image DBN.
3. Repeat step 2 till queue is empty.
4. Form the CPTs for the image DBN by suitably marginalizing the original CPTs.

Table 4.1. Algorithm for finding simple projection homomorphisms assuming that the MDP is completely specified.

general "structured projection" is one where each feature of S' is computed as a function of a subset of features of S, and the subsets corresponding to each feature of S' are disjoint (Zeigler, 1972). Without suitable constraints, often derived from prior knowledge of the structure of the problem, searching for generalized structured projections results in a combinatorial explosion. Boutilier and colleagues have investigated other forms of structured morphisms assuming various representations of the CPTs of structured MDPs—when the morphism is defined by boolean formulae of the features (Boutilier and Dearden, 1994), when it is defined by decision trees on the features (Boutilier et al., 1995), and when it is defined by first-order logic formulae (Boutilier et al., 2001).

4.2.3 Permutation Symmetry Groups

It can be shown that symmetry groups do not result in structured projection homomorphisms, except in a few degenerate cases. A simple class of structured morphisms that do lead to useful symmetry groups are those generated by *permutations* of feature values. Let Σ_M be the set of all possible permutations of $\{1, \dots, M\}$. Given a structured set $X \subseteq \prod_{i=1}^{M} X_i$ and a permutation $\sigma \in \Sigma_M$, we can define a permutation on X by $\sigma(\langle x_1, \dots, x_M \rangle) = \langle x_{\sigma(1)}, \dots, x_{\sigma(M)} \rangle$, and it is a *valid* permutation on X if $x_{\sigma(i)} \in X_i$ for all i and for all $\langle x_1, \dots, x_M \rangle \in X$.

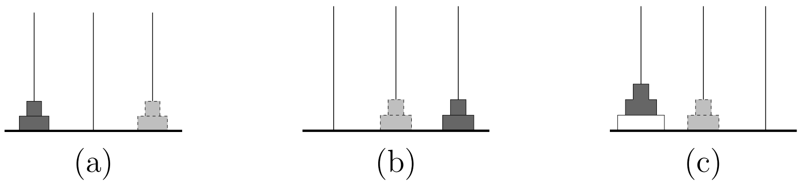


Figure 4.4. Towers of Hanoi. The task is to move the disks from the darker position to the lighter position. Panels (a), (b) and (c) show situations that are equivalent under permutation symmetries.

Definition: A *permutation automorphism* h on a structured MDP $\langle S, A, \Psi, P, R\rangle$ is a bijection on Ψ defined by a tuple of bijections $\langle f, \{g_s | s \in S\}\rangle$, with $h(s,a) = (f(s), g_s(a))$, where $f \in \Sigma_M : S \to S$ is a valid permutation on S, and $g_s : A_s \to A'_{f(s)}$ for $s \in S$, such that:

$$\begin{array}{rcl} P(f(s), g_s(a), f(s')) & = & P(s, a, s'),\ \forall s, s' \in S, a \in A_s \\ & = & \displaystyle\prod_{i=1}^{M} \text{Prob}(s'_{f(i)} | f(\text{Parents}_{f(s)}(s'_{f(i)}, a)) \\ R(f(s), g_s(a)) & = & R(s,a),\ \forall s \in S, a \in A_s \end{array}$$

Here $f(\text{Parents}_{f(s)}(s'_i, a)) = \{s_{f(j)} | s_j \in \text{Parents}(s'_i, a)\}$ with $s_{f(j)}$ assigned according to $f(s)$.

Permutation symmetries arise often in checking correctness of concurrent process models, multi-agent systems and especially of interest to us, in environments with objects in them, as in the *towers of Hanoi* problem (Figure 4.4). The symmetric MDP shown in Figure 1.2(a) also has a permutation symmetry. If the states of the MDP is described by the x and y co-ordinates, then exchanging these features and action N with E and S with W is an automorphism on the MDP. This together with the identity map, gives rise to the reflectional symmetry discussed earlier. Permutation symmetries arise in such environments due to object interchangeability—when you can exchange some objects in the world with other similar objects. This is represented as a permutation in which the features corresponding to a object are permuted with that of another object. If this permutation is part of a symmetry group, then the objects are interchangeable.

In general, a permutation symmetry group, $\mathcal{G}$, does not give rise to an $\mathcal{G}$-reduced MDP that requires fewer features than the original to describe the state set. Therefore permutation symmetry groups do not necessarily yield a more compact description of the problem, even if there are fewer states in the reduced MDP than in the original. Except in some degenerate cases it is difficult to stipulate conditions under which there is a reduction in the size of the feature set. A straight forward approach to minimization using permutation symmetry groups would require us to enumerate all the state-action pairs of the MDP. Even given a symmetry group, $\mathcal{G}$, constructing the quotient MDP by explicitly enumerating all the state-action pairs of the MDP, takes time proportional to $|\Psi| \cdot |\mathcal{G}|$.

Table 4.2 presents a more efficient incremental algorithm for constructing the quotient MDP given a symmetry group or subgroup. This algorithm constructs the quotient MDP directly without first generating the induced partition. Here we need to examine only one representative from each block of the partition induced by the symmetry group. Since applying a permutation to the feature values takes time linear in the number of features, this algorithm could run in time polynomial in the number of features and the size of Ψ' in the best case. While in the worst case this algorithm might take time proportional to $|\Psi|$, it will frequently run much faster than that. This is an adaptation of an algorithm proposed by Emerson and Sistla (1996) for constructing quotient models for concurrent systems. The algorithm as presented assumes that all states of the MDP are reachable from any starting state. It is easy to modify the algorithm to cases in which this is not true.

```
Given $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ and $\mathcal{G} \leq \mathrm{Aut}\mathcal{M}$,
construct $\mathcal{M}/B_{\mathcal{G}} = \langle S', A', \Psi', P', R' \rangle$.
Set $Q$ to some initial state $\{s_0\}$, $S' \leftarrow \{s_0\}$
While $Q$ is non-empty
    s = dequeue(Q)
    For all $a \in A_s$
        If $(s, a) \not\equiv_{\mathcal{G}} (s', a')$ for some $(s', a') \in \Psi'$, then
            $\Psi' \leftarrow \Psi' \cup (s, a)$
            $R'(s, a) = R(s, a)$
            For all $t$ such that $P(s, a, t) > 0$
                If $t \equiv_{\mathcal{G}|S} s'$, for some $s' \in S'$,
                    $P'(s, a, s') \leftarrow P'(s, a, s') + P(s, a, t)$
                else
                    $S' \leftarrow S' \cup t$
                    $P'(s, a, t) = P(s, a, t)$
                    add $t$ to $Q$.
```

Table 4.2. Incremental algorithm for constructing the $\mathcal{G}$-reduced image given MDP $\mathcal{M}$ and some $\mathcal{G} \leq \mathrm{Aut}\mathcal{M}$. Q is the queue of states to be examined. This algorithm terminates when at least one representative from each equivalence class of $\mathcal{G}$ has been examined.

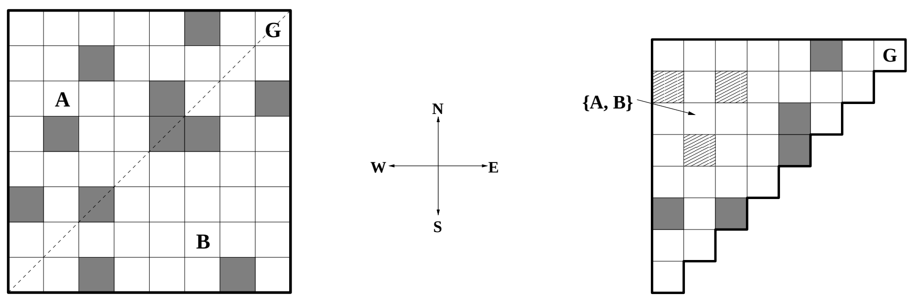
4.2.4 Discussion

The structured morphisms we introduced in this section leverage the fact that the state sets are structured and that the transition probabilities can be factored and expressed as largely independent components. In order to further exploit the structure in the transition dynamics we would employ more structure representations of the CPT such as decision trees. Some of the existing work that operate with such structured representations can be viewed as methods for incrementally determining homomorphic projections. But such methods do not explicitly model the process as a minimization approach. Givan et al. (2003) explore some methods that explicitly exploit structure in minimization, but more work is needed in this direction. The simple structured morphisms we introduced here are nevertheless powerful enough to achieve reductions in a variety of problem domains, as illustrated by the examples earlier. In fact, many of the homomorphic reductions we consider later in the thesis are in this class of structured morphisms. This class of morphisms is especially useful when we consider partial reductions and reductions in a hierarchical setting. For example, as we shall demonstrate later, Dietterich (2000a) considers the class of simple projections in his work on safe-state abstraction in the MaxQ hierarchical framework.

4.3 Approximate Equivalence

The MDP homomorphism conditions on which we base our notions of equivalence are very strong conditions and are satisfied only in some restricted classes of problems. Nevertheless in practice we frequently encounter problems for which we can derive useful "approximate" reduced models by employing relaxed notions of equivalence. We construct these approximate models by aggregating together states and actions that differ slightly in their dynamics. For example, consider the gridworld shown in Figure 4.5(a). This is a slightly modified version of the symmetric grid world from Figure 1.2(a). While the MDP is more or less symmetric about the NE-SW diagonal as before, there are a few states including A and B that are not symmetric. These differences do not affect the optimal policy for reaching the goal significantly, and we can form a reduced MDP (Figure 4.5(b)) which is similar to the MDP shown in Figure 1.2(b). Here we need to treat the lightly shaded states differently, since these are non-symmetric states.

In this section we introduce two concepts to model this approximate minimization. An *approximate homomorphism* uses the average behavior of the aggregated states and is particularly useful in learning, while a *bounded homomorphism* employs Bounded-parameter MDPs (Givan et al., 2000) and allows us to derive bounds on the loss of performance resulting from the approximation.



(a) (b)

Figure 4.5. (a) A slightly asymmetric gridworld problem. The goal state is G and there are four deterministic actions. The problem is approximately symmetric about the NE-SW diagonal. (b) A reduced model of the gridworld in (a). The state-action pairs (A, E) and (B, N) in the original problem both correspond to the pair $(\{A, B\}, E)$ in the reduced problem. A solution to this reduced gridworld can be used to derive an approximate solution to the full problem.

4.3.1 Approximate Homomorphisms

In many circumstances we can aggregate together states and actions that have slightly different dynamics to form a reduced model. The most straight forward choice for the dynamics of the reduced model is an average, possibly a "weighted" average, of the dynamics of the state-action pairs that belong to the same equivalence class. In the absence of additional knowledge about the problem, a useful heuristic is to consider a simple average of the aggregated dynamics. Formally we define an approximate homomorphism as follows:

Definition: An *approximate MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\}\rangle$, with $h((s, a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that for all s, s' in S and $a \in A_s$:

$$P'(f(s), g_s(a), f(s')) = \frac{1}{\left|[(s,a)]_{B_h}\right|} \sum_{(q,b)\in[(s,a)]_{B_h}} T(q, b, [s']_{B_h|S}) \quad (4.4)$$

$$R'(f(s), g_s(a)) = \frac{1}{\left|[(s,a)]_{B_h}\right|} \sum_{(q,b)\in[(s,a)]_{B_h}} R(q, b). \quad (4.5)$$

We call $\mathcal{M}'$ the *approximate homomorphic image* of $\mathcal{M}$ under h. To determine the transition probability $P'(f(s), g_s(a), f(s'))$ in $\mathcal{M}'$ we first compute the block transition probability from each element of $[(s,a)]_{B_h}$ to the block $[s']_{B_h|S}$. Then we set the transition probability to be the average of these block transition probabilities. Note that if h is a homomorphism, then the induced partition satisfies the SSP property and each of the block transition probabilities we compute above are equal to one another. We do a similar computation for the reward function as well.

When we employ such approximate reduced models to do planning or learning, the appropriate aggregate dynamics to employ is a weighted average of the dynamics of the state-action pairs that belong to a given equivalence class, the weights being determined by the frequency with which each member of the class is encountered in the course of the solution process. As we shall see in later chapters, while learning with online experience it is sufficient to specify only the state, action and reward spaces of the image MDP and the trajectories through state-action space the agent experiences implicitly induce the transition probabilities. In such cases, the induced transition probabilities of the image MDP will account for the frequency of visitation.

Example of an Approximate Homomorphism

Consider the MDP shown in Figure 4.6(a). This represents a spatial navigation task. The goal is to reach the set of shaded states in the center of the environment. The darker regions are obstacles, while the clear regions are open space. Consider the four quadrants formed by the dotted lines in the figure, it is clear that the environment is more or less symmetric. An approximate homomorphic image of this MDP can be formed as shown in Figure 4.6(b). Once again, the clear regions are open space and the dark regions like C are obstacles. The lightly shaded regions like A and B use aggregate dynamics as described above. In particular, if we assume that the original

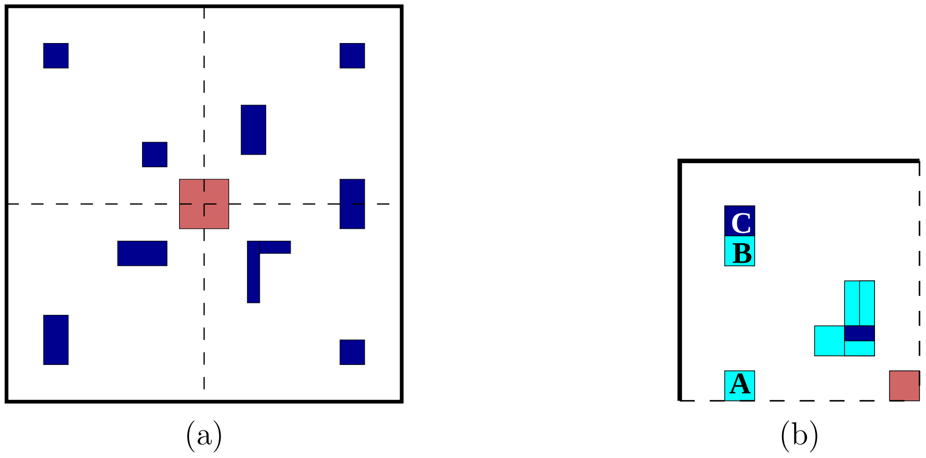


Figure 4.6. (a) A spatial navigation problem. The goal is to reach the shaded region in the center. The environment is approximately symmetric about the dotted lines. (b) An approximate homomorphic image of the task in (a). Transitions into the lightly shaded regions are determined either by aggregate dynamics or specified as intervals. See text for more details.

problem was deterministic, then the probability of being able to move into region A is one half and that of being able to move into region B is three quarters.

4.3.2 Bounding the Approximation Loss

With the relaxation of the homomorphism conditions we lose some of the guarantees we established earlier. In particular the optimal value equivalence theorem is no longer guaranteed to hold. A policy that is optimal in an approximate homomorphic image is not necessarily optimal when lifted to the original MDP. But if the approximation is a reasonable one, the lifted policy is not too far from the optimal. We would like to bound the "distance" between the true optimal policy and the policy lifted from the image. We do this by deriving an upper limit on the maximum difference between the optimal value function in the original MDP and the value function of the lifted policy.

We adopt results from Whitt (1978) to derive this bound. Whitt explored the issue of approximation and abstraction in the *contraction mapping* formulation of a dynamic program due to Denardo (1967). This formulation is a generalization of MDPs, stochastic games and other sequential decision making paradigms. Whitt explores the issue of approximating a dynamic program from the point of optimal value preservation and considers state-action equivalence, along with state-action value functions. He derives precise conditions for when an image accurately captures the optimal values of the original dynamic program and also looks at sequence of approximations that in the limit converge to an exact image. He also derives bound on the loss in the optimal value function when the image is an approximation. He specializes some of the results to stochastic sequential decision problems, from which we can derive the equivalent results for MDPs. More details of his results are in Appendix B. The bounds depend on the differences in the resulting aggregate parameters and the actual parameters. Let $h = \langle f, \{g_s | s \in S\} \rangle$ be an approximate homomorphism from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to an MDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$. We define the following quantities:

$$\begin{aligned}
K_r &= \max_{\substack{s \in S \\ a \in A_s}} | \, R(s,a) - R'(f(s), g_s(a)) \, | \\
K_p &= \max_{\substack{s \in S \\ a \in A_s}} \sum_{[s_1]_f \in B_f} \left| T(s, a, [s_1]_f) - P'(f(s), g_s(a), f(s_1)) \right| \\
\delta_{r'} &= \max_{\substack{s' \in S' \\ a' \in A_{s'}}} R'(s', a') - \min_{\substack{s' \in S' \\ a' \in A_{s'}}} R'(s', a')
\end{aligned}$$

where K_r is the maximum difference between the aggregate block reward and the actual reward, K_p is the maximum difference between the actual block transition probabilities and the aggregate transition probabilities and δ_r' is the range of the reward function in the image MDP. Then the following theorem holds:

Theorem 8: Let $\pi'^{\star}$ be an optimal policy in $\mathcal{M}'$ and $\pi'^{\star}_M$ be that policy lifted to $\mathcal{M}$. Let γ be the discount factor. Then:

$$\left\| V^{\star} - V^{\pi'^{\star}_M} \right\| \leq \frac{2}{1-\gamma} \left(K_r + \frac{\gamma}{1-\gamma} \delta_{r'} \frac{K_p}{2} \right)$$

.

Proof: Let $h = \langle f, \{g_s | s \in S\} \rangle$ be the approximate homomorphism from $\mathcal{M}$ to the image $\mathcal{M}'$. Let $V^{\pi'^\star}$ be the optimal value function in $\mathcal{M}'$. Let $\tilde{V}^{\pi'^\star}$ be the function constructed by *lifting* $V^{\pi'^\star}$ to $\mathcal{M}$, i.e., $\tilde{V}^{\pi'^\star}(s) = V^{\pi'^\star}(f(s))$. Note that $\tilde{V}^{\pi'^\star}$ is not necessarily the same function as $V^{\pi'^\star_M}$ since h is only an approximate homomorphism. Let

$$Q(s, a, V) = R(s, a) + \gamma \sum_{s' \in S} P(s, a, s')V(s'),$$

for some real valued function V on S. From Theorem 6.1 of Whitt (1978) stated in Appendix B, we have the following:

$$K(V^{\pi'^\star}) = \max_{\substack{a \in A_s \\ s \in S}} \left| Q(s, a, \tilde{V}^{\pi'^\star}) - Q(f(s), g_s(a), V^{\pi'^\star}) \right| \leq K_r + \frac{\gamma}{1-\gamma} \delta_{r'} \frac{K_p}{2}.$$

Since $\mathcal{M}$ and $\mathcal{M}'$ are MDPs, we employ corollary (b) of Theorem 6.1 here. From the corollary to Lemma 3.1 we have:

$$\left\| V^\star - V^{\pi'^\star_M} \right\| \leq \frac{2}{1-\gamma} K(V^{\pi'^\star}).$$

Since $V^{\pi'^\star}$ is optimal in $\mathcal{M}'$, we omit the terms that arise due to deviation from optimality of the value function. We get the desired result put combining the above two results. $\square$

Here the distance between the value functions is measured using the max-norm, i.e., $\|V_1 - V_2\| = \max_{s \in S} |V_1(s) - V_2(s)|$. Thus Theorem 8 allows us to bound the maximum deviation from the true optimal function that results from using a particular approximate homomorphic image. This result holds only for values of $\gamma < 1$. When γ is 1, it is possible to construct examples where the error is unbounded. For small value of γ the overall error depends more on the difference in the immediate reward, since the second term within the parenthesis is small. This is not surprising, since small γ leads to more myopic optimal policies. Similarly for large values of γ the error depends more on the differences in the transition probabilities and the range of the rewards function, since these quantities affect the long term return.

While the derivation of the bound does not depend on the details of the averaging method used, the bounds themselves could vary, since how we average influences the values of K_p, K_r and $\delta_{r'}$. Therefore these bounds can be computed beforehand if we are using simple averages or some fixed weighted averaging scheme. If we want to use the visitation frequencies in order to derive our reduced MDPs, we need to dynamically recompute our bounds as we gather more information. In order to avoid this we look at another formulation of approximate homomorphisms.

4.3.3 Bounded Approximate Homomorphisms

Approximate homomorphisms provide a convenient tool for deriving approximate homomorphic images. When we do not have access to the visitation frequencies, the uniform average formulation we have adopted is somewhat unsatisfying. We chose this formulation in the absence of additional information as per Occam's razor. While this is not a issue while learning, as pointed out earlier, it is something that needs to be addressed in certain situations like planning domains and when we try to derive bounds on the loss in asymptotic performance a priori.

Dean et al. (1997) addressed the same problem in the context of their model minimization framework. They developed the concept of a *Bounded-parameter MDP* (BMDP) (Givan et al., 2000) that allows one to model the differences in the block transition probabilities of the aggregated states. A BMDP is an MDP in which the transition probabilities and the rewards are specified as intervals. Formally a BMDP $\mathcal{M}'$ is given by the tuple $\langle S, A, \Psi, P_{\updownarrow}, R_{\updownarrow} \rangle$ where S and A are the state and action sets, Ψ is the set of admissible state-action pairs, $P_{\updownarrow} : \Psi \times S \to [0,1] \times [0,1]$ with $P_{\updownarrow}(s,a,s') = [P_{low}(s,a,s'), P_{high}(s,a,s')]$, for all (s,a) in Ψ and s' in S, is the range of values for the probability of transiting from s to s' under action a and $R_{\updownarrow} : \Psi \rightarrow \mathbb{R}\times\mathbb{R}$, with $R_{\updownarrow}(s,a) = [R_{low}(s,a), R_{high}(s,a)]$, for all (s,a) in Ψ, is the range of the expected reward on performing action a in state s. We extend their work to our abstraction framework and develop the notion of a bounded approximate homomorphism, that gives us an alternate way of modeling the reduction shown in Figure 4.6.

Definition: A *bounded approximate MDP homomorphism* h from an MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to a BMDP $\mathcal{M}' = \langle S', A', \Psi', P'_{\updownarrow}, R'_{\updownarrow}\rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\}\rangle$, with $h((s,a)) = (f(s), g_s(a))$, where $f : S \rightarrow S'$ and $g_s : A_s \rightarrow A'_{f(s)}$ for $s \in S$, such that, $\forall s, s' \in S$ and $a \in A_s$:

$$P'_{\updownarrow}(f(s), g_s(a), f(s')) = \left[\min_{t\in[s]_{B_h|S}} T(t, a, [s']_{B_h|S}), \max_{t\in[s]_{B_h|S}} T(t, a, [s']_{B_h|S})\right] \quad (4.6)$$

$$R'_{\updownarrow}(f(s), g_s(a)) = \left[\min_{t\in[s]_{B_h|S}} R(t,a), \max_{t\in[s]_{B_h|S}} R(t,a)\right] \quad (4.7)$$

When aggregating states with slightly different block transition probabilities, we set the transition probabilities in the image to be a range covering the maximum and minimum of these block transition probabilities. As with an approximate homomorphism, if the block transition probabilities do not differ, then the above definition reduces to that of a MDP homomorphism. In the example in Figure 4.6(b) if we model the reduced MDP as the image of a bounded approximate homomorphism, then the probability of being able to move into any of the lightly shaded regions is given by the range $[0,1]$. The clear regions indicate open space and the darkly shaded regions obstacles as before.

Note that a bounded parameter MDP actually specifies a family of MDPs, each member of which has transition probabilities and rewards drawn from the ranges specified subject to the constraint that each row of P sum to 1. For example, if we use aggregate dynamics to determine the parameters of the reduced MDP is Figure 4.6(b) it is then a member of the family of BMDPs described by the bounded approximate

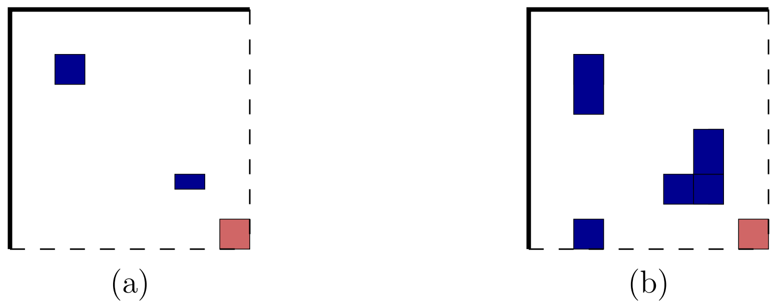


Figure 4.7. (a) Optimistic choice of parameters for the BMDP shown in Figure 4.6(b). (b) Pessimistic choice of parameters for the same BMDP. In both the figures, dark squares are obstacles and the goal is to reach the shaded area in the lower right. See text for explanation of parameter choices.

homomorphism. More generally, for any form of averaging we use in an approximate MDP homomorphism the resulting image MDP is a member of this family.

The *interval value iteration* algorithm of Givan et al. (2000) allows us to derive bounds for the optimal value function in a BMDP. These bounds can then be used to bound the loss of performance that arises due to employing bounded approximate homomorphisms. The basic idea behind interval value iteration is that at each time step we assume that for the next iteration we pick parameters from the allowable range that will give us the best (worst) possible returns to derive upper (lower) bounds on the optimal policy. To illustrate this idea, let us return to the MDP in Figure 4.6(b). The upper bound for the value function is given by solving the optimistic MDP shown in Figure 4.7(a). Here all the light squares are considered as open space. The lower bound is given by solving the pessimistic MDP shown in Figure 4.7(b). Here all the light squares are considered obstacles.

The bounded optimal value function does not always tell us what the optimal policy should be, such as in cases when the ranges of possible next state values overlap. But these bounds do give us some intuition into whether the approximate homomorphism we are using is a reasonable one. For example if the lower bound tells us that for every policy in the MDP there is some selection of the parameters that renders the problem impossible to solve, then we cannot use the current reduction profitably.

4.3.4 Discussion

The definition of an approximate homomorphism we introduced here is very inclusive. In fact, it is possible to define an approximate homomorphism from any MDP to any other arbitrary MDP. But the bounds given by Theorem 8 will be very loose and there is no practical utility in defining such homomorphisms. An useful approximate homomorphism should be one in which the values of K_r and K_p are "reasonable". One measure of usefulness is to check if the loss in performance that results from lifting the solution of the approximate image to the original problem is acceptable. In Chapters 5 and 6 we will see examples of such "acceptable" approximate homomorphisms.

Theorem 8 gives an upper bound on the loss of performance due to the approximation. The lower bound even when we use an approximate homomorphic image is zero. In other words, it is still possible to recover the optimal solution of the original problem by lifting the solution of an approximate homomorphic image. Such a situation arises due to the fact that optimal policies when defined as acting greedily with respect to the optimal value functions are sensitive only to the relative ordering of the values. In fact, we need to identify correctly only the action with the highest value in each state. This is possible in many situations when we apply approximate homomorphic images and enhances the utility of such images. In Chapter 5 we see an example of such an approximate homomorphic image.

Givan et al. (2000) developed the notion of a BMDP while studying approximate minimization in the Dean and Givan minimization framework. They base their work on related formulations of *MDPs with imprecise parameters* (e.g., Satia and Lave, 1973; White and Eldeib, 1986, 1994) from operations research. Apart from developing BMDPs and the interval value iteration algorithm, Givan et al. also investigate in detail the question of constructing reduced BMDP models of given MDPs. They conclude that it is not usually possible to specify a unique reduced BMDP model, and that we have to resort to some heuristic to choose between several equally viable alternatives. They also show that we cannot guarantee that there is a heuristic which in all cases will lead to the best possible BMDP, i.e., one that gives the best bounds on the value functions and the smallest reduced models. The utility of the bounded approximate homomorphism formulation is as a tool for deriving a priori bounds, loose bounds in many cases, on the loss in performance when we employ a particular abstraction.

Whitt (1978) uses a notion of approximation similar to our definition of approximate homomorphism. His motivation was to develop an abstraction framework for dynamic programs. He outlines a method to derive homomorphic images of dynamic programs by successively refining the approximate image so that the error bounds become tighter. Kim and Dean (2001) have developed a similar method for MDPs. They also successively construct better approximate images of a given MDP, but the criterion they use to refine the image is the actual performance of the image MDPs' solutions when lifted to the original MDP. We believe that this is a more promising direction for developing iterative algorithms to finding minimal images of an MDP.

CHAPTER 5
ABSTRACTION IN HIERARCHICAL SYSTEMS

One of the significant recent advances in RL has been the introduction of temporal abstraction frameworks and hierarchical learning algorithms (Sutton et al., 1999; Dietterich, 2000a; Parr and Russell, 1997). Such frameworks allow us to systematically ignore decisions at fine time scales and employ "temporally-extended" actions that let us operate at coarser time scales. Humans routinely employ temporal abstraction. For example, consider the problem of getting a cup of coffee from a vending machine. A typical plan would be "go to coffee machine, feed the machine change and get coffee". One does not plan at the level of individual steps or muscle twitches or neuronal signals. The above policy can then be used for getting coffee as part of a still higher level plan, say "get bagel, get coffee, go to conference room and start meeting". Being able to learn and reason at multiple temporal scales dramatically widens the applicability of RL to large-scale, complex systems.

As discussed in Section 3.7, MDP homomorphisms can be readily employed by RL algorithms for spatial abstraction in flat, or non-hierarchical systems modeled as MDPs. Hierarchical organization of the learning architecture provides us with additional opportunities for abstraction. One can consider abstractions specific to a particular sub-problem in the hierarchy or to a family of sub-problems. In the representation of higher level tasks, redundancy introduced by suitably defined lower level problems in the hierarchy can be exploited. The lower level problems hide the small differences in the one-step transition dynamics allowing us to capture higher level task structure. The notion of MDP homomorphism can be extended to a convenient and powerful formalism for modeling abstraction schemes in hierarchical systems also.

While the popular hierarchical frameworks differ in many of the details, they have one thing in common: all of them model the hierarchical system as a family of semi-Markov decision processes (SMDPs). Extending the notion of MDP homomorphism to SMDPs allows the modeling of abstractions at various levels of a hierarchy using equivalence notions similar to those developed in Chapter 3.

Typically, sub-problems at different levels of a hierarchy are defined over a subset of the state-action space of the original problem. To enable the modeling of abstractions in such sub-problems we introduce the notion of a "partial homomorphism". Informally a partial homomorphism is a surjection from an MDP, or an SMDP, to a corresponding image, such that the homomorphism conditions hold only over subsets of the state-action space. This notion is very useful when considering sub-task specific abstraction and also in developing a hierarchical task decomposition framework that extends the *options* framework (Sutton et al., 1999).

5.1 SMDP Homomorphisms

Recall from Chapter 2 that a discrete-time semi-Markov decision process (SMDP) is a generalization of an MDP in which actions can take variable amounts of time to complete. Specifically, an SMDP is a tuple $\langle S, A, \Psi, P, R \rangle$, where S, A and Ψ are the sets of states, actions and admissible state-action pairs; $P : \Psi \times S \times \mathbb{N} \rightarrow [0, 1]$ is the transition probability function with $P(s, a, s', N)$ being the probability of transition from state s to state s' under action a in N time steps and $R : \Psi \times \mathbb{N} \rightarrow \mathbb{R}$ is the expected discounted reward function, with $R(s, a, N)$ being the expected reward for performing action a in state s and completing it in N time steps. Traditionally SMDP transitions are modeled using two separate distributions—one for the next states and one for the transition, or holding, times. We are adopting the formalism of Dietterich

(2000a) since it is more suitable for modeling hierarchical RL frameworks. When the SMDP has well defined terminal states, the future rewards are often not discounted. In such cases an SMDP is equivalent to an MDP and the transition times can be ignored.

The natural generalization of an MDP homomorphism to SMDPs is as follows:

Definition: An *SMDP homomorphism* h from an SMDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ to an SMDP $\mathcal{M}' = \langle S', A', \Psi', P', R' \rangle$ is a surjection from Ψ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\} \rangle$, with $h((s, a)) = (f(s), g_s(a))$, where $f : S \to S'$ and $g_s : A_s \to A'_{f(s)}$ for $s \in S$, such that $\forall s, s' \in S, a \in A_s$ and for all $N \in \mathbb{N}$:

$$\begin{aligned} P'(f(s), g_s(a), f(s'), N) &= \sum_{s'' \in [s']_f} P(s, a, s'', N), & (5.1) \\ R'(f(s), g_s(a), N) &= R(s, a, N). & (5.2) \end{aligned}$$

As before $\mathcal{M}'$ is called a homomorphic image of $\mathcal{M}$. Most of the results developed in Chapter 3 for MDP homomorphisms hold for SMDP homomorphisms. The homomorphism conditions, as discussed in Section 3.8, are rarely satisfied exactly in practice and relaxed notions of equivalence have to be considered to obtain useful abstractions. It is doubly difficult in the case of SMDPs to satisfy the homomorphism conditions exactly since the transition times are also considered. One useful way to relax the equivalence notion, analogous to the earlier definition of approximate homomorphism, is to allow small variations in the transition times and assign a suitable weighted average to the corresponding transitions in the image SMDP. Note that since we are considering transitions that take multiple time steps, errors tend to accumulate, and Theorem 8 on error bounds for approximate homomorphisms does not apply. However, it is possible to derive similar, although looser, error bounds for suitably defined relaxations of SMDP homomorphisms. But as we shall see later in this chapter and the next, useful abstractions of SMDPs can be modeled using MDP homomorphisms, completely ignoring transition times. Dietterich (2000a) argues that such abstractions are valid in finite horizon problems, where the discount factor can be set to 1. They are also valid in settings where any solution achieved quickly is more desirable than a costly search for the optimal solution.

5.2 Partial Homomorphisms

One of the chief obstacles to using abstraction approaches based on homomorphic equivalence is that often there exists no surjection from one MDP to another such that both conditions of a homomorphism hold for the entire Ψ space of the MDP. Even in such cases it is sometimes possible to derive useful abstractions by restricting attention to a subset of Ψ. For example, consider the problem of navigating in a grid world like environment shown in Figure 5.1(a). Note that when the dark square is the goal, the entire gridworld is homomorphic to the image shown in Figure 5.1(b). If the goal is moved to one of the lighter squares, this is no longer true. In fact, it is not possible to come up with non-trivial homomorphic images in these cases. But, regardless of the position of the goal, it is possible to define a "partial" morphism from the gridworld to the image shown in Figure 5.1(c), so that the homomorphism conditions holds for the states in the room. All other state-action pairs are mapped to a special absorbing state-action pair in the image, indicated by a dark oval and a solid arrow.

Partial homomorphisms may also be formed by restricting the actions over which the homomorphism conditions hold. This is especially useful in environments with objects, where classes of objects would behave similarly under some set of actions while not under others. For example, if the action under consideration is hitting a nail, then both a hammer and a shoe behave similarly, while they are very dissimilar in general. An analogous situation would be defining homomorphisms over only a subset of actions. Formally, a partial homomorphism is defined as follows:

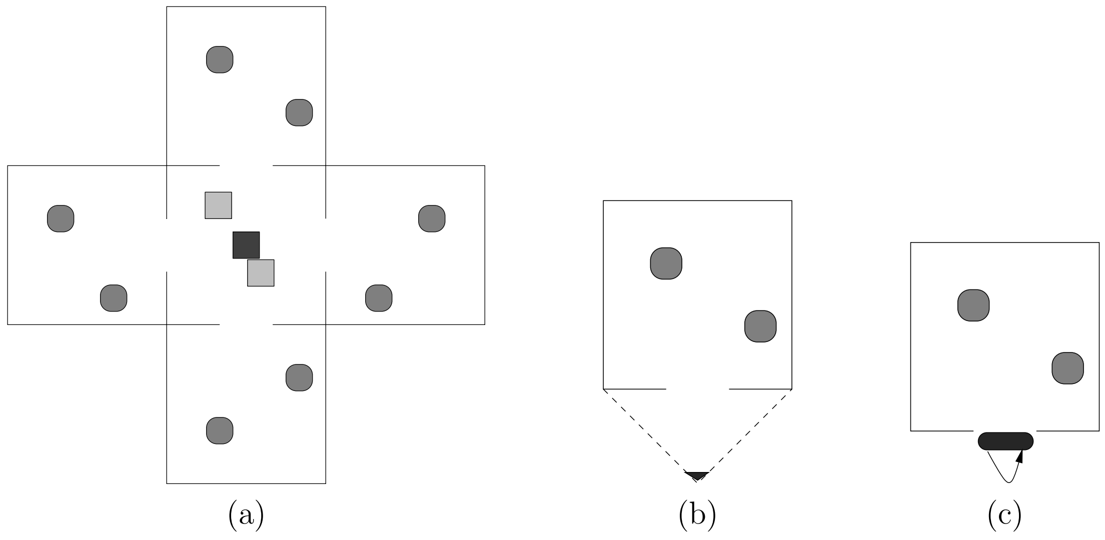


Figure 5.1. (a) A gridworld task with *rooms* and the usual gridworld dynamics. The dark square indicates the goal. The lighter squares are alternate locations for the goal. (b) A homomorphic image when the dark square is the goal. The goal in this image is the dark triangle at the bottom. The transitions wrap around the dotted lines, i.e., actions W and S at the left edge will cause transitions to the right edge and action E and N at the right edge cause transitions to the left edge. (c) A partial homomorphic image restricted to the *room states*. The dark oval is an absorbing state.

Definition: A *partial MDP homomorphism* from $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ to $\mathcal{M}' = \langle S', A', \Psi', P', R'\rangle$, such that $\tau \in S'$, $\alpha \in A'$, $(\tau, \alpha) \in \Psi'$ and $P'(\tau, \alpha, \tau) = 1.0$, is a surjection from $\Upsilon \subseteq \Psi$ to Ψ', defined by a tuple of surjections $\langle f, \{g_s | s \in S\}\rangle$, with $h(s, a) = (f(s), g_s(a))$, where $f : S \to S'$ and $g_s : \Upsilon_s \to A'_{f(s)}$ for $s \in S$ where Υ_s is non-empty and given by $\Upsilon_s = \{a | (s, a) \in \Upsilon\}$, such that for all $s \in f^{-1}(S' - \tau)$, $s' \in S$ and $a \in \Upsilon_s$:

$$P'(f(s), g_s(a), f(s')) = T(s, a, [s']_{B_h|S}) \tag{5.3}$$

$$R'(f(s), g_s(a)) = R(s, a). \tag{5.4}$$

$\mathcal{M}'$ is called the *partial homomorphic image* of $\mathcal{M}$ under h. Partial SMDP homomorphisms can be similarly defined with the conditions above extended to hold for joint distributions of next state and transition times. The state τ is an absorbing state in $\mathcal{M}'$ with one action α that transitions to τ with probability 1. The homomorphism conditions hold only in states that do not map to τ. All the actions in states that map to τ, map to α. Lifting policies defined in $\mathcal{M}'$ yield policy fragments in $\mathcal{M}$, with action probabilities specified only for elements in the *support* of h, i. e., $\Upsilon = h^{-1}(\Psi' - (\tau, \alpha))$. Similarly, the support of f is the subset of S given by $f^{-1}(S' - \tau)$. In the example in Figure 5.1, τ corresponds to the state represented as a black oval in Figure 5.1(c) and α is indicated by the solid arrow. All state-action pairs, with the state component in the central hall, map to (τ, α) under the partial homomorphism. If the task in the image MDP is treated as an episodic task, then an optimal way to exit the room can be learned.

The above definition of a partial homomorphism facilitates the development of the following material on hierarchical problem decomposition. In practice the exact form of the above definition is seldom required. Partial homomorphisms are usually employed in modeling abstraction in a particular sub-problem in a hierarchy. As we shall see shortly, the description of the sub-task typically circumscribes the state and action sets. Hence one can define homomorphisms that hold only over these restricted sets, which when viewed with respect to the original MDP are partial homomorphisms.

5.3 Sub-goal Options

In this work, the hierarchical framework we adopt is the *options framework* introduced by Sutton et al. (1999). While the ideas developed here are more generally applicable, we chose the options framework for the flexibility it offers.

In the *options framework*, in addition to the "primitive" actions that are part of the problem definition, the learning agent can employ temporally extended actions or *options*. For example, in addition to primitive actions such as, move one step north, south, east or west, we would consider "options" such as *get coffee*, *go to conference room*, etc., as additional actions. Formally, an option (Sutton et al., 1999) in an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is defined by the tuple $O = \langle \mathcal{I}, \pi, \beta \rangle$, where the initiation set $\mathcal{I} \subseteq S$ is the set of states in which the option can be invoked, π is the policy to be followed while the option is executing, and the termination function $\beta : S \rightarrow [0, 1]$ gives the probability of the option terminating in any given state. The option policy can in general be a mapping from arbitrary sequences of state-action pairs (or histories) to action probabilities. This allows us to model option policies that are dependent of time, for example, *do action a_1 10 times.* The set of states over which the option policy is defined is known as the *domain* of the option. The option policy might map to other options as well as primitive actions, in which case the option is called a *hierarchical option.* An MDP with options is naturally modeled as an SMDP with the transition time distributions induced by the option policies. See Precup (2000) for more details on deriving the SMDP parameters and on learning models of options.

While the options framework offers several advantages such as simplicity of representation and flexibility in the hierarchical structure, it does not address several key issues relating to hierarchical task decomposition. Specifically the framework assumes that the option policies are fixed and does not address the question of learning simultaneously at multiple levels of the hierarchy. Precup (2000) suggests a method for learning the policies for a class of options, but it is largely an offline method and does not address the issue of online learning. Other researchers have used standard RL algorithms (McGovern and Barto, 2001; Jonsson and Barto, 2001) to learn option policies in specific problem settings by but have not explored a general solution. This issue is of particular interest to us, since our abstraction ideas not only lead to more compact representations of the problem (and hence the policies) but also to more efficient solution methods. So to take full advantage of the abstract representation we want to learn the option policies as well as the solution to the original problem.

We introduce a modification of the options framework which we call sub-goal options, that facilitates learning at multiple levels of the hierarchy simultaneously and also allows us to employ abstractions that are specific to a particular level in the hierarchy. We consider the class of options whose policies satisfy the Markov property and terminate on achieving a sub-goal. In such instances it is possible to implicitly define the option policy as the solution to an *option MDP*. We adopt the following definition of a sub-goal option:

Definition: An *sub-goal option* of an MDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is defined by $O = \langle \mathcal{M}_O, \mathcal{I}, \beta \rangle$, where $\mathcal{M}_O = \langle S_O, A_O, \Psi_O, P_O, R_O \rangle$ is the option MDP, $\mathcal{I} \subseteq S$ is the initiation set of the option, and $\beta : S_O \rightarrow [0, 1]$, is the termination function.

The set S_O is a subset of S and constitutes the domain of the option, A_O is a subset of A, and the reward function, R_O, is chosen to reflect the sub-goal of the option. The transition probabilities, P_O, are induced by P. The option policy π is obtained by solving $\mathcal{M}_O$, treating it as an episodic task with the possible initial states of the episodes given by $\mathcal{I}$ and the termination of each episode determined by β.

Figure 5.2 shows an example of a Markov sub-goal option. The task in this domain is to gather the objects in each of the rooms. The task is described in greater detail later in the chapter. For the time being, consider the sub-task of collecting the object in Room 1. An option can be defined to achieve this, using the option MDP shown in Figure 5.2(b). The states in the MDP are the cells in Room 1 along with a boolean variable indicating possession of the object. The reward function is +1 on exiting the room with the object, and 0 otherwise. The initiation set is all the cells in Room 1 and β is set to 0 in the room and 1 elsewhere. The option policy is given by the optimal policy in this MDP.

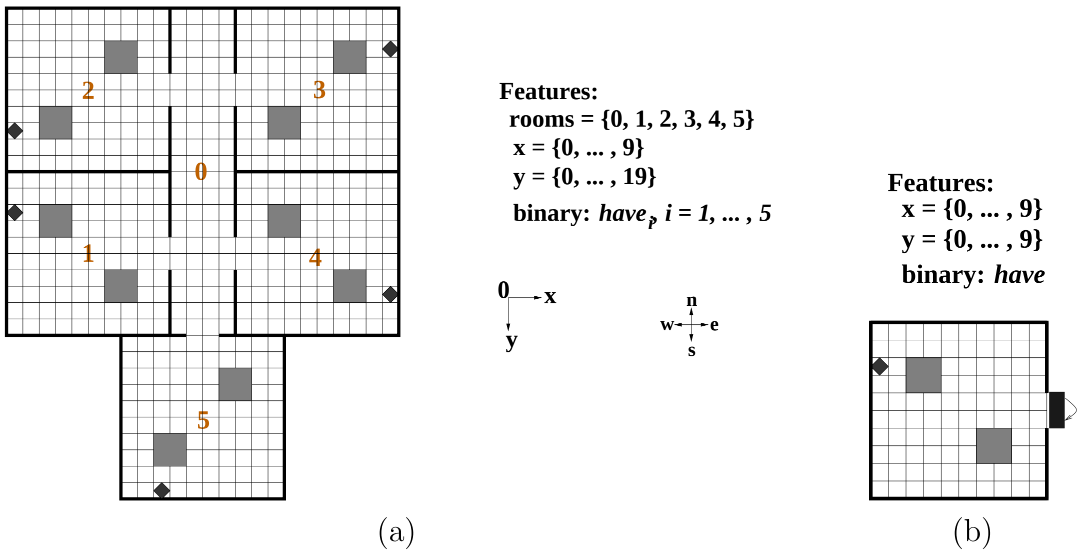


Figure 5.2. (a) A simple rooms domain with similar rooms and usual stochastic gridworld dynamics. The task is to collect all 5 objects (black diamonds) in the environment and reach the central corridor. The shaded squares are obstacles. (b) The option MDP corresponding to a *get-object-and-leave-room* option. See text for full description.

Sub-goal options model sub-tasks whose policies map to only primitive actions. Correspondingly one can define hierarchical sub-goal options with policies mapping to other options as well as primitive actions.

Definition: A *hierarchical sub-goal option* is given by the tuple $O = \langle \mathcal{M}_O, \mathcal{I}, \beta \rangle$, where $\mathcal{M}_O = \langle S_O, A_O, \Psi_O, P_O, R_O \rangle$ is the *option SMDP*, and $\mathcal{I}$ and β are as defined earlier.

The set S_O is the domain of the option and A_O contains other options as well as primitive actions. As before, the reward function R_O is chosen to reflect the sub-goal of the option. The policies of the lower level options influences the transition probabilities P_O. Hence, to derive P_O, it is assumed that the lower level options are following fixed policies which are optimal in the corresponding option SMDPs.

Sub-goal options are only a special class of options and are not as inclusive as the original definition of an option. But this class covers a wide range of useful options and more importantly, the policies of sub-goal options are easily learned using standard RL approaches. While policies for Markov options that represent continuing tasks and do not have a well defined sub-goal are also easy to learn, the utility of such options in a hierarchical architecture is not clear. Such options do not yield control to higher levels of the hierarchy. So except when such an option is at the root level of the hierarchy, indefinitely continuing execution is not a desirable property for an option in a hierarchical setting.

The above definition of a Hierarchical sub-goal option associates a SMDP with the option. Partial SMDP homomorphisms can now be employed to model abstractions specific to an option. In the next section we develop a formal mechanism for employing abstractions in option SMDPs.

5.4 Relativized Options

An option SMDP $\mathcal{M}_O$ can be expressed as a partial homomorphic image of the MDP $\langle S, A, \Psi, P, R_O \rangle$. To formally model $\mathcal{M}_O$ as a partial homomorphic image, we add an absorbing state τ to S_O, an absorbing action α to A_O and (τ, α) to Ψ_O. Now the partial homomorphism $h = \langle f, \{g_s | s \in S\} \rangle$ from $\mathcal{M}$ to $\mathcal{M}_O$ is defined as follows:

$$f(s) = \begin{cases} s, & \text{if } s \text{ is in the domain of the option} \\ \tau, & \text{otherwise.} \end{cases}$$

$$g_s(a) = \begin{cases} a, & \text{if } s \text{ is in the domain of the option and } (s, a) \in \Psi_O \\ \alpha, & \text{if } s \text{ is not in the domain of the option.} \end{cases}$$

$g_s(a)$ is not defined for $(s, a) \in \Psi$ such that s is in the domain of the option and (s, a) is not in Ψ_O. h is referred to as the option homomorphism corresponding to option O. The equivalence classes of Ψ induced by h are mostly singletons, except for the pre-image of (τ, α). Thus this partial homomorphism does not result in any useful abstraction of the original state space. In many cases it is possible to define partial homomorphisms on the MDP $\langle S, A, \Psi, P, R_O \rangle$ with non-trivial equivalence classes. By suitably modify the definition of a sub-goal option, the resulting image MDP can be used as an option MDP along with the corresponding homomorphism, allowing option specific abstractions. The structure and redundancy not present over the entire problem but present only when considering a sub-task can then be exploited in forming abstractions. Before formally extending the definition of a sub-goal option, let us look at another implication of using homomorphic images as option MDPs.

Again consider the problem of navigating in the gridworld environment shown in Figure 5.2(a). The goal is to reach the central corridor after collecting all the objects in the environment. The main task is naturally broken into several sub-tasks, the goal of each is to collect the object and exiting from a room. One could define 5 sub-goal options to model each of these sub-tasks as discussed in the previous section. However these sub-tasks are very similar to each other and in fact the option MDPs of the corresponding sub-goal options are isomorphic to one another. This similarity can be exploited to define a single partial homomorphism from the original MDP to any of the option MDPs, one of which is shown in Figure 5.2(b). Employing such an abstraction gives rise to a compact representation of a related family of options, in this case the tasks of collecting objects and exiting each of the five rooms, using a single option MDP. This compact sub-goal option is referred to as a *relativized option*. Such abstractions are an extension of the notion of relativized operators introduced by Iba (1989). Formally we define a relativized option as follows:

Definition: A *relativized option* of an SMDP $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is the tuple $O = \langle h, \mathcal{M}_O, \mathcal{I}, \beta \rangle$, where $\mathcal{M}_O = \langle S_O, A_O, \Psi_O, P_O, R_O \rangle$ is the option SMDP, $\mathcal{I} \subseteq S$ is the initiation set, $\beta : S_O \to [0, 1]$ is the termination function and $h = \langle f, \{g_s | s \in S\} \rangle$ is a partial homomorphism from the SMDP $\langle S, A, \Psi, P, R_G \rangle$ to $\mathcal{M}_O$ with R_G chosen to describe the sub-goal.

The set S_O is the image of the domain of the option under f plus an absorbing state τ, and $\Psi_O = h(\Psi)$. The option policy $\pi : \Psi_O \rightarrow [0, 1]$ is obtained by solving $\mathcal{M}_O$ by treating it as an episodic task as before. Depending on the sub-task, h can be defined only over a subset of Ψ, restricting the actions and options available in the option SMDP.

The option policy π now encodes the policy for all the related sub-tasks in the original problem that map onto the option SMDP. Going back to our example in Figure 5.2(a), we can now define a single *get-object-and-leave-room* relativized option using the option MDP of Figure 5.2(b). The policy learned in this option MDP can then be suitable lifted to $\mathcal{M}$ to provide different policies in the different rooms. Thus, if the optimal action in a particular state in the image MDP is E, it is lifted to give E in Rooms 1 and 2, W in Rooms 3 and 4 and N in Room 5.

5.5 Hierarchical Problem Decomposition

Relativized options allow us to model a variety of abstract representations. As described in the previous section, even a "regular" sub-goal option, i.e., one that does not employ any abstraction, can be defined as a relativized option where the option homomorphism is given by the identity map on the domain of the option and a map to (τ, α) elsewhere. Given that relativized options facilitate hierarchy specific abstractions, it is particularly desirable that we learn the option policies online, since we can considerably speed up learning performance as we shall demonstrate shortly. Although the options framework allows us great flexibility in specifying hierarchies, it does not explicitly address the question of simultaneously learning at multiple levels of the hierarchy.

In order to learn policies at different levels of the hierarchy we first need to specify a suitable decomposition of the learning problem. We develop a hierarchical problem decomposition approach, similar to MAXQ decomposition (Dietterich, 2000a), based on relativized options. The decomposition divides the learning problem into several components— one component each for learning the various relativized option policies and one root component for learning to solve the original task using the various option policies.

Suppose we are given an SMDP $\mathcal{M}$ whose action set contains a set $\mathcal{O}$ of relativized options, $O_i = \langle h_i, \mathcal{M}_i, \mathcal{I}_i, \beta_i \rangle$, $i = 1, \cdots n$. The relativized options may call other options in $\mathcal{O}$, subject to the constraint that there are no loops in the resulting call graph. This implicitly encodes a hierarchy, with the options whose action sets consist of only primitive actions being at the lowest level.

Definition: The *hierarchical decomposition* of $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is given by $\{O_0, O_1, \cdots, O_n\}$, where $O_0 = \langle h_0, \mathcal{M}_0, \mathcal{I}_0, \beta_0 \rangle$, is a relativized option describing the *root task*, with $\mathcal{M}_0$ a homomorphic image of $\mathcal{M}$ under h_0. $\mathcal{I}_0$ is the set of start states and β_0, the termination function indicating the set of terminal states for the original task.

Note that while $\mathcal{M}_i$ for $i > 0$ are partial homomorphic images of $\mathcal{M}$ with the reward function replaced by the suitable option reward, $\mathcal{M}_0$ is a homomorphic image of $\mathcal{M}$ with the original reward function. If all the actions and options in A are considered while defining O_0 it frequently results in a very inefficient decomposition in which solving the root task is equivalent to solving the entire problem. In such cases the root task is defined by using a partial homomorphic image of an MDP formed from $\mathcal{M}$ by restricting the homomorphism to some subset of Ψ. In other words, certain options and actions are only allowed to be chosen in a smaller set of states than in which they are admissible.

Consider the example in Figure 5.2(a). A single relativized option *get-object-and-leave-room* which is admissible in all states in the rooms, including the doorways, can be defined. The primitive actions are admissible everywhere. If the root task is formed by minimizing this MDP, the reduced task is the same as the original task. Instead only the relativized option is allowed to be picked in the states in the room. Remember that the option terminates on exiting the room. Thus, when it is invoked in a room, there are four possible states the option can cause a transition to—the two corridor cells adjacent to the doorway, with or without the object. The probabilities of transitioning to these states varies with the location the options were started in the room. Thus to achieve useful reductions, there is a need to consider approximate homomorphisms. With suitable relaxations of the homomorphism criterion, an approximate image of the original task can be formed, which consists of only the hallway and doorway states, with only the relativized option admissible in the doorway states. The hierarchical decomposition for this example is given by the root task described by the above approximate image and the *get-object-and-leave-room* option.

For the above example it is clear that some prior knowledge about the structure of the task is needed before finding a suitable hierarchical decomposition. This has been the bane of all hierarchical RL frameworks. Autonomously finding such decompositions has been the focus of recent work by McGovern and Barto (2001) and Hengst (2002), but a universal task decomposer is unlikely to be developed in the near future.

5.5.1 Hierarchical Policies

Definition: A *hierarchical policy* in $\mathcal{M}$ with a hierarchical decomposition of $\{O_0, \cdots, O_n\}$ is specified by the tuple $\pi = \langle \pi_0, \pi_1, \cdots, \pi_n \rangle$, i.e., a policy for each O_i. An *optimal hierarchical policy*, $\pi^\star$, consists of the tuple of optimal policies in each of the component options, i.e., $\pi^\star = \langle \pi_0^\star, \pi_1^\star, \cdots, \pi_n^\star \rangle$.

Our definition of a hierarchical sub-goal option assumes that the lower level options are following a fixed policy that is optimal in their respective option SMDPs. Therefore the above notion of optimality is equivalent to that of *recursive optimality* (Dietterich, 2000a). A recursively optimal hierarchical policy is one in which policies at

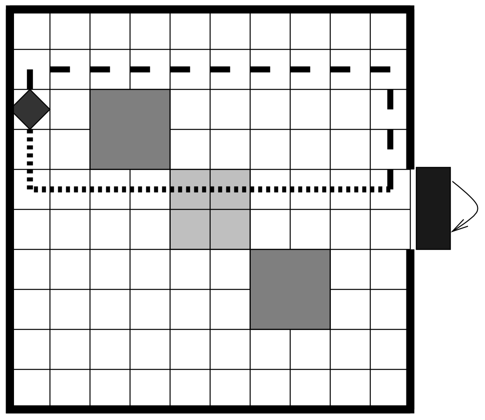


Figure 5.3. The modified option MDP corresponding to the *get-object-and-leave-room* relativized option. The lightly shaded squares in the middle of the room indicate states with a negative reward. The dashed line indicates the optimal policy for this modified option MDP. The dotted line indicates a component of the hierarchically optimal policy, when using this option in the task described by Figure 5.2

each level of the hierarchy is optimal given that the policies of all the lower level tasks are (recursively) optimal. A recursively optimal hierarchical policy is not necessarily the optimal policy of an MDP.

The hierarchical decomposition and the associated call graph restrict the class of representable policies to only a subset of all possible policies over Ψ. Optimality in this restricted space can be defined analogous to the definition for flat MDPs—a hierarchical policy is said to be *hierarchically optimal* is it uniformly dominates all other hierarchical policies consistent with the given hierarchical decomposition. Under this definition lower level option policies need not be optimal in the corresponding option MDP. Recursive optimality is a still weaker form of optimality. Thus recursively optimal policies are not guaranteed to be hierarchically optimal.

Returning to the example in Figure 5.2. Suppose that the sub-task modeled by the option is modified by introducing a negative reward in the 4 states in the middle of the room (Figure 5.3). Now the optimal policy for this option is to avoid these squares and take a more circuitous route (dashed line in Figure 5.3). Thus, a hierarchical policy that is recursively optimal will use that policy for this option. But for the overall goal of gathering all the objects, the policy that goes straight to the goal is better (dotted line in Figure 5.3). Thus a hierarchically optimal policy will use that policy for this option.

A hierarchically optimal policy usually yields a better performance on a particular problem instance, while a recursively optimal policies lead to better option policies. Learning better option policies is a desirable property when we are learning to solve many related problems instead of a single instance of a problem. Therefore we adopt recursive optimality as the optimality criterion in this work. Dietterich (2000a) introduced these notions of optimality to the RL community and discussed the issue of hierarchical versus recursive optimality in more detail.

The hierarchical decomposition framework is similar to the MAXQ task decomposition framework (Dietterich, 2000a). MAXQ does allow simultaneous learning at all levels in the hierarchy, but imposes a more rigid hierarchical structure on the possible policies. MAXQ employs a form of value function decomposition in which the optimal value function of a task is constructed as a combination of the optimal value function of its children in the call graph. This is a compact representation of the value function and is an essential part of the MAXQ-Q-learning algorithm. This requires that the children maintain a value function that reflects the task objectives of their parent. Dietterich uses a pseudo-reward function in addition to the task reward function to specify the objectives of a sub-task. Thus, MAXQ requires that the sub-task policies be optimal with respect to a combination of the sub-task objectives and the root task objectives. This can in some cases inhibit sub-task sharing and reuse, since changes in the root task's objectives make the sub-task non-optimal. Since we focus only on learning optimal policies in our framework, one consequence of adopting our approach is that we cannot always recover the optimal value function in all the states for our root task. If the root task is described by a homomorphism restricted to a subset of

Ψ, the optimal value function of the root task will not be defined for the part of Ψ over which the homomorphism is not defined. In order to recover the optimal value function for all elements of Ψ we would need to formulate our framework to use a MAXQ like value function decomposition. But we are not exploring that direction in this thesis.

5.5.2 Learning with Hierarchical Decompositions

The simplest choice for a learning algorithm with our hierarchical decomposition framework is to use Q-learning for learning the lowest level option policies and SMDP Q-learning at the higher levels. Dietterich (2000b) calls this *hierarchical SMDP Q-learning*. By arguments similar to that used by Dietterich (2000a) it can be shown that the following result holds:

Theorem 9: Let $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$, be an episodic SMDP with well defined terminal states and discount factor γ. Let $\{O_0, O_1, \cdots, O_n\}$ be a hierarchical decomposition of $\mathcal{M}$. Let $\alpha_t(i) > 0$ be a sequence of constants for each option i such that

$$\lim_{T\to\infty} \sum_{t=1}^{T} \alpha_i(t) = \infty \quad \text{and} \qquad \lim_{T\to\infty} \sum_{t=1}^{T} \alpha_i^2(t) < \infty.$$

Let the exploration policy in each option be a GLIE policy (Jaakkola et al., 1994), i.e., one such that: (i) each action/option is chosen infinitely often in each state during learning and (ii) in the limit of infinite exploration they become greedy with respect to the value function. Then with probability 1 hierarchical SMDP Q-learning converges to a recursively optimal policy for $\mathcal{M}$ consistent with the given hierarchical decomposition.

Proof: (sketch) This is a restatement of Theorem 3 from Dietterich (2000a). The proof follows an argument similar to those introduced to prove the convergence of Q-learning (Bertsekas and Tsitsiklis, 1996). The conditions on the learning rates, $\alpha_t(i)$, are required for the convergence of Q-learning and SMDP Q-learning. The reason the exploration policy needs to satisfy the above conditions is the following. Consider an option in the hierarchy which in turn calls other options. If the lower level option continues to execute a non-greedy exploratory policy, the dynamics perceived by the higher level corresponds to that policy and not to the true optimal policy of the option. Hence learning in the higher level option does not converge to a recursively optimal solution. Given the above conditions the proof proceeds by induction from the options at the lowest level to the root level task. □

One consequence of Theorem 9 is that now learning can proceed simultaneously at all levels of the hierarchical decomposition—the higher levels do not need to wait until the lower levels converge before they begin learning. All that is required is that the learning in the lower levels eventually converge to their optimal policies and we are still guaranteed to converge overall to a recursively optimal policy. In all the hierarchical learning results reported in this work we employ our hierarchical problem decomposition framework with hierarchical SMDP Q-learning.

The question that naturally arises in this setting is how would one define option homomorphisms if the lower level option policies are not known apriori. We answer this criticism along similar lines as Dietterich (2000a). Often the designer has prior knowledge of the structure of the problem and can identify some subgroup of the symmetry group or a partial homomorphism to employ in minimization. For example, in navigation tasks, regardless of the policies of the lower level tasks, it is possible to define spatial symmetries. In Chapter 6 we present a experimental design in a control framework in which a viable hierarchical control structure is available due to prior design. In cases where this is not possible, we can employ online abstraction ideas such as Jonsson and Barto (2001) to refine our option MDP homomorphism as learning progresses. Such an approach would start from a very approximate homomorphic image and successively refine the approximation till we obtain a satisfactory model.

5.6 Illustrative Example

We now provide a complete description of the simple gridworld task in Figure 5.2(a) and some experimental results to illustrate the utility of relativized options and our hierarchical decomposition. The agent's goal is to collect all the objects in the various rooms by occupying the same square as the object. Each of the rooms is a 10 by 10 grid with certain obstacles in it. The actions available to the agent are $\{N, S, E, W\}$ with a 0.1 probability of *failing*, i.e., going randomly in a direction other than the intended one. This probability of failing is referred to as the *slip*.

The state is described by the following features: the room number the agent is in, with 0 denoting the corridor, the x and y co-ordinates within the room or corridor with respect to the reference direction indicated in the figure and boolean variables $have_i$, $i = 1, \ldots, 5$, indicating possession of object in room i. Thus the state with the agent in the cell marked A in the figure and having already gathered the objects in rooms 2 and 4 is represented by $\langle 3, 6, 8, 0, 1, 0, 1, 0\rangle$. The goal is any state of the form $\langle \cdot, \cdot, \cdot, 1, 1, 1, 1, 1\rangle$ and the agent receives a reward of +1 on reaching a goal state.

We compared the performance of an agent that employs relativized options with that of an agent that uses multiple regular options. The "relativized" agent employs a single relativized option, O_r, whose policy can be suitably lifted to apply in each of the 5 rooms. The relativized option MDP corresponds to a single room and is shown in Figure 5.2(b). The state space S' of the option MDP is defined by 3 features: x and y co-ordinates and a binary feature *have*, which is true if the agent has gathered the object in the room. There is an additional absorbing state-action pair (τ, α), otherwise the action set remains the same. The stopping criterion β is 1 at τ and zero elsewhere. The initiation set consists of all states of the form $\langle i, *\rangle$, with $i \neq 0$. There is a reward of +1 on transiting to τ from any state of the form $\langle *, 1\rangle$, i.e. on exiting the room with the object. One can see that lifting a policy defined in the option MDP yields different policy fragments depending on the room in which the option is invoked. For example, a policy in the option MDP that picks E in all states would lift to yield a policy fragment that picks W in rooms 3 and 4, picks N in room 5 and picks E in rooms 1 and 2.

The "regular" agent employs 5 regular options, $O_1, \cdots, O_5$, one for each room. Each of the option employs the same state space and stopping criterion as the relativized option. The initiation set for option O_i consists of states of the form $\langle i, * \rangle$. There is a reward of $+1$ on exiting the room with the object. Both agents employ SMDP Q-learning Bradtke and Duff (1995) at the higher level and Q-learning Watkins (1989) at the option level.

In both cases the root task, O_0 is described as follows: The state set of $\mathcal{M}_0$ is described by the the room number the agent is in, the various $have_i$ features and if the agent is in the central corridor, then the x and y co-ordinates of the agent; the admissible actions are the primitive actions in the corridor and the corresponding options in the room doorways; the transition and reward functions are those induced by the original task and the option policies. The initiation set is the set of states in the corridor with all $have_i$ features set to false. The termination condition is 1 for states in the corridor with all $have_i$ features set to true. It is 0 elsewhere.

We also compared the performance of an agent that employs only the four primitive actions. All the agents used a discount rate of 0.9, learning rate of 0.05 and ϵ-greedy exploration, with an ϵ of 0.1. The results shown are averaged over 100 independent runs. The trials were terminated either on completion of the task or after 3000 steps. Figure 5.4(a) shows the asymptotic performance of the agents. This a hard problem for the primitive action agent and it takes around 30,000 iterations before it learns a reasonable policy and another 15,000 before it even approaches optimality. This is often the case when employing RL on even moderately large problems and is one of the chief reason for choosing a hierarchical approach. Since we are more interested in comparing the performance of the option agents, we do not present further results for the primitive action agent. In fact in some of the later tasks, the primitive action agent does not learn to solve the task in any reasonable amount of time.

Figure 5.4(a) also demonstrates that the option agents perform similarly in the long run, with no significant difference in performance. This demonstrates that there is no loss in performance due to the abstractions we employ here. This is not surprising since the homomorphism conditions are met exactly in this domain.

Figure 5.4(b) shows the initial performance of the option agents. As expected, the relativized agent significantly outperforms the regular agent in the early trials[1]. Figure 5.5 graphs the rate at which the agents improved over their initial performance. The relativized agent achieved similar levels of improvement in performance significantly earlier than the regular option. For example, the relativized agent achieved a 60% improvement in initial performance in 40 trials, while the regular agent needed 110 trials. These results demonstrate that employing relativized options significantly speeds up initial learning performance, and if the homomorphism conditions hold exactly, there is no loss in the asymptotic performance.

Employing a hierarchical approach results in a huge improvement in performance over the primitive action agent. While there is a significant improvement in performance while employing relativized options, this is not comparable the initial improvement over primitive actions. One might ask is this improvement worth the additional expense of relativizing the options. Our answer to this two fold. First, the relative magnitudes of improvement is an artifact of this problem domain. In more complex domains, with more redundancy a greater improvement in performance is to be expected. In many cases employing some form of an hierarchy is the only feasible approach and in such cases we can obtain further improvement in performance for

[1]All the significance tests were *two sample t-tests* with a p-value of 0.01.

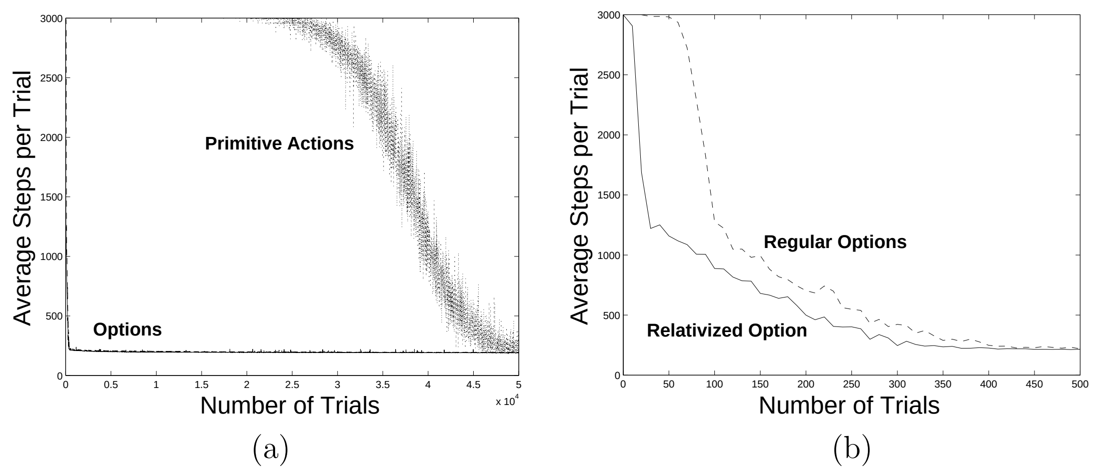


Figure 5.4. (a) Comparison of asymptotic performance of various learning agents on the task shown in Figure 5.2. See text for description of the agents. (b) Comparison of initial performance of the regular and relativized agents on the same task.

some additional cost by relativization. Second, using relativized option opens up the possibility of being able to train an agent to perform a sub-task in some prototypical environment. Once the agent acquires a reasonable policy in this training task then it is able to generalize to all instance of this task. This is particularly useful if training experience is expensive, for example in the case of real robotic systems.

5.7 Approximate Equivalence

The various rooms in the test bed above map exactly onto the option MDP in Figure 5.2(b). In practice, such exact equivalences do not arise often. To study the usefulness of relativized options in inexact settings, we conducted further experiments in which the rooms had different dynamics. In the first task, the rooms had the same set of obstacles, but had different slips. In the corridor actions fail with probability 0.1 and in rooms 1 through 5 with probabilities 0.2, 0.3, 0.25, 0.5 and 0.0, respectively. The relativized option in this case used an approximate homomorphic image, with the transition probabilities of the option MDP being determined by a weighted average of the various slips, the weights being determined by how often the rooms were visited in

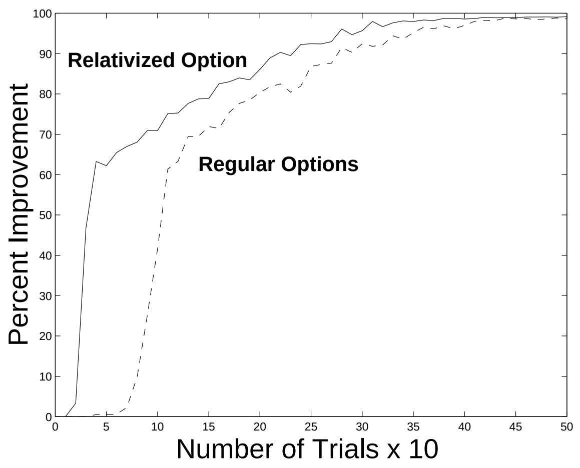
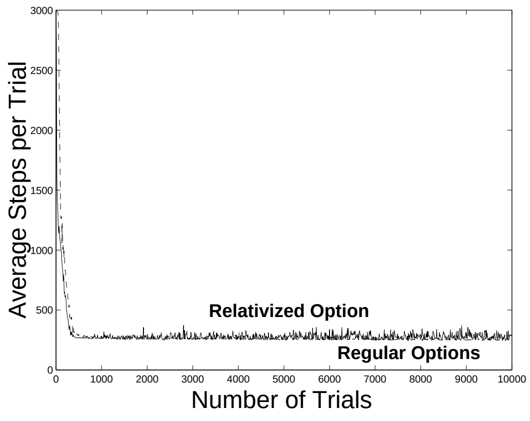


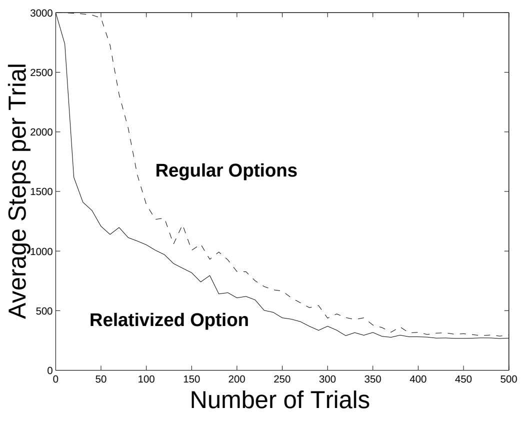
Figure 5.5. Comparison of the rate of improvement to final performance of the regular and relativized agents on the task shown in Figure 5.2.

a trial. When the learning converges the probabilities are determined by just a simple average of the slips. Figure 5.6(b) shows the initial performance of the relativized agent and the regular agent on this task. Again the relativized agent significantly outperforms the regular agent initially and the asymptotic performance, Figure 5.6(a), shows no significant difference.

In the second task, the rooms have differently shaped obstacles, as shown in Figure 5.7(a). The relativized option uses an approximate homomorphic image whose transition probabilities were determined by the various obstacle locations and visitation frequencies of each room. Again there is a significant improvement in initial performance, but the asymptotic performance of the relativized agent is slightly, but significantly, worse than the regular agent, as shown in Figures 5.8(a) and 5.8(b). This loss in asymptotic performance is expected and is observed in other inexact scenarios we tested the agents on. This loss was not observed in the previous example since there was a single policy that was optimal in all the rooms and in the option MDP, despite the various slips. In some cases the loss due to approximation reaches



(a)

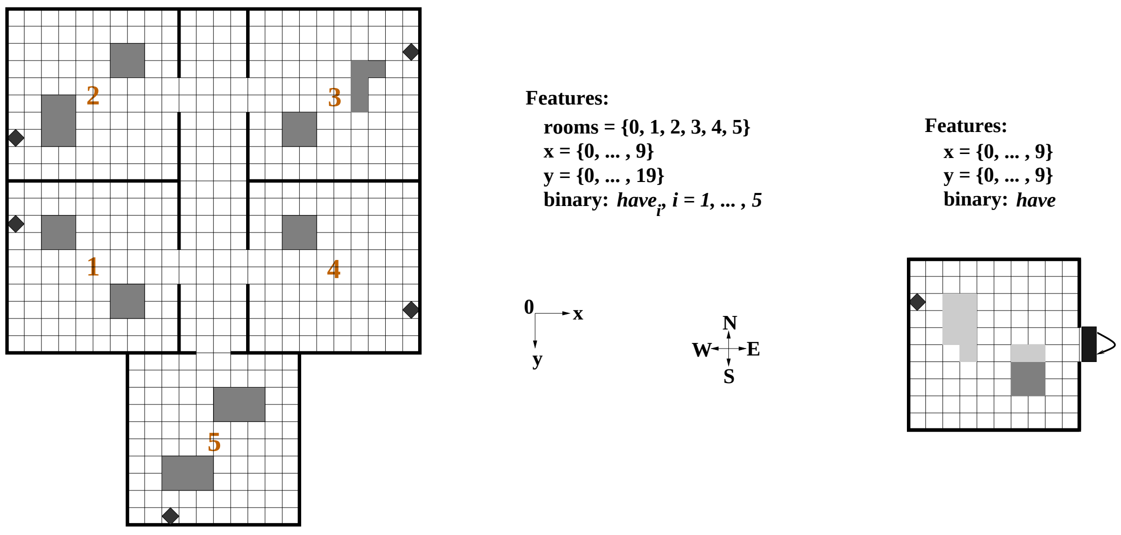


(b)

Figure 5.6. (a) Comparison of asymptotic performance of the regular and relativized agents on the modified rooms task. See text for description of the task. (b) Comparison of initial performance of the two agents on the same task.

unacceptable levels, with the relativized agent failing to successfully complete the task on certain trials even after considerable training.

This loss can be bounded by modeling the relativized option using a bounded approximate homomorphism from each of the rooms to the image BMDP shown in Figure 5.7(b), where the probabilities of transitioning into the lightly colored states range from 0 to 1. We can now use interval value iteration to bound the range of the optimal value function. The tasks on which the agent fails would have zero lower bounds that indicate that the task cannot be solved. In this example such a scenario would arise if the obstacles in the various rooms were placed in such a way that there was a chain of light and dark colored cells across the length of the room, cutting of access to the object from the doorway. In the case shown in Figure 5.7(b) the lower bound for the optimal value for the doorway states is γ^{28}, pessimistically assuming that all the light colored cells are obstacles. The upper bound is γ^{24}, 24 being the length of the shortest path to the goal, assuming that all the light colored cells are clear.



(a) (b)

Figure 5.7. (a) A simple rooms domain with dissimilar rooms. The task is to collect all 5 objects in the environment. (b) The option BMDP corresponding to a *get-object-and-leave-room* option. See text for full description.

5.8 Relation to MAXQ Safe State-abstraction

Dietterich (2000a) introduced safe state-abstraction conditions for the MAXQ architecture. He states a few intuitive "rules" to ignore certain features of the state space. These conditions ensure that the resulting abstractions do not result in any loss of performance. This was the first systematic study of abstraction in hierarchical RL frameworks. He established that performing abstraction using these rules does not change the value function learned, since there is no loss of necessary information. Many of the conditions for safe abstraction are applicable only to the MAXQ value function decomposition and to the particular form of MAXQ task decomposition. The following condition is more universal and applies to the our hierarchical decomposition framework as well:

Definition: A Projection ρ_J is *safe* if: (i) for all (s, a) in Ψ and s' in S, $P(s, a, s', N) = Prob(\rho_J(s'), N|\rho_J(s), a) \times Prob(\rho_{M-J}(s')|s, a)$, and (ii) for all $(s, a), (t, a)$ in Ψ, if $\rho_J(s) = \rho_J(t)$, then $R(s, a, N) = R(t, a, N)$.

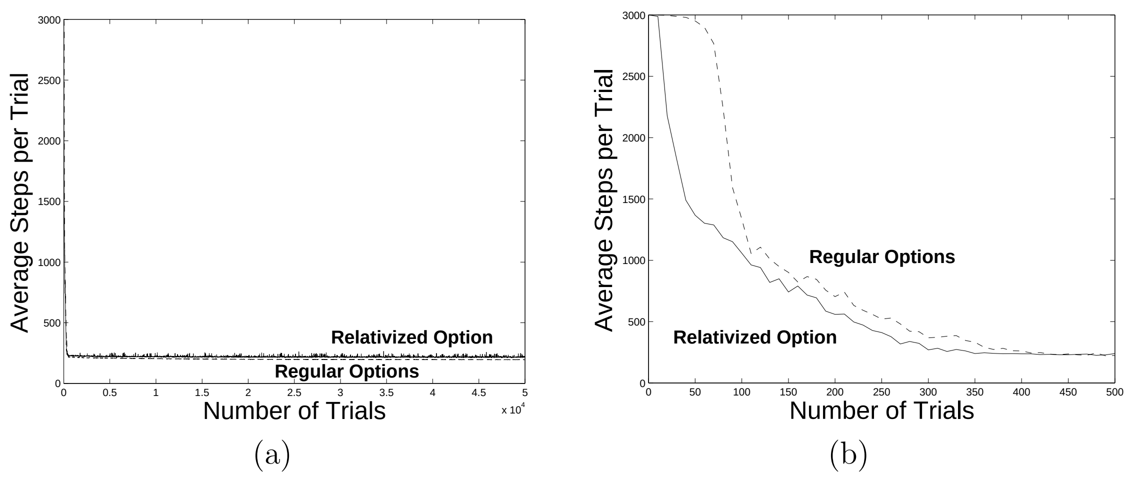


Figure 5.8. (a) Comparison of asymptotic performance of the regular and relativized agents on the task in Figure 5.7. (b) Comparison of initial performance of the two agents on the same task.

Condition (i) states that the transition probability can be expressed as a product of two probabilities, one of which describes the evolution of the subset of the features describing the abstract state space and depends only on that subset. Condition (ii) states that if two states project to the same abstract state, then they have the same immediate reward. From our earlier definitions of projection and SMDP homomorphisms (Equations 4.2, 4.3, 5.1 and 5.2), it is evident that a safe ρ_J is also a projection homomorphism. Thus, the above conditions are equivalent to the SMDP homomorphism conditions restricted to simple projection homomorphisms and not considering action remapping. Thus, the SMDP homomorphism conditions generalize Dietterich's safe state-abstraction condition as applicable to the our hierarchical problem decomposition.

5.9 Related Work

Hierarchical architectures enable hiding low level details, but much of recent work on hierarchical RL has focused on hiding the temporal scale and not the spatial details. To the best of our knowledge, ours is the first work to apply minimization ideas and symmetry based reductions to hierarchical RL. We present our approach using the options framework, but our ideas are equally applicable to the other major hierarchical RL frameworks—MaxQ (Dietterich, 1998) and hierarchy of abstract machines (HAMs) (Parr and Russell, 1997). In the MaxQ framework, the given MDP is decomposed into a hierarchy of sub-MDPs and each MDP is treated as a sub-task, that might employ the solutions of the MDPs at lower levels. The options framework with Markov options that follow our definition is very close to the MaxQ framework. But MaxQ imposes a rigid hierarchy on the various sub-MDPs, while the options framework allows for a lot of flexibility in the formation of hierarchies during learning.

While model minimization as such has not been applied to hierarchical RL, there has been some work in hierarchy specific abstraction. We mentioned Dietterich's (2000c) results on "safe" abstraction in Section 6. He states a few intuitive "rules" that allow us to ignore certain features of the state space. Some of the rules are general, for example, ignore certain features at a particular level if they do not affect the value function at that level. Other rules are specific to the MaxQ framework. He establishes that performing abstraction using these rules does not change the value function learned, since there is no loss of necessary information. Andre and Russel (2001b) presents similar results for the HAM framework. While Dietterich's general rules apply to the options framework as well, their application to options has not been studied in detail.

Jonsson and Barto (2001) develop an algorithm for automatically forming option specific abstractions. Their algorithm is based on McCallum's (1995) *U-trees*, a decision tree based automatic abstraction scheme for POMDPs. The U-trees algorithm performs abstraction by using a decision tree over histories of experience and makes sufficient distinctions so as to represent the value function accurately. Jonsson and

Barto apply the algorithm to building option specific state abstractions, by modifying the definition of histories to include information about the option being executed. Hernandez-Gardiol and Mahadevan (2001) propose a hierarchical POMDP model that at different levels of the hierarchy, considers histories that consist of decisions made at that level. This too is a hierarchy specific abstraction scheme, but one tailored to a POMDP setting.

Programmable HAMs (PHAMs) (Andre and Russel, 2001a) are an extension to the HAM framework that among other things allow us to define parameterized HAMs. For the purposes of this discussion we can assume that a HAM is equivalent to an option with partly specified policies and during learning the agent needs to fill in the unspecified parts of the policy. In the PHAM framework, a family of HAMs can be specified as a single parameterized HAM, with the HAM policy also depending on the value of the parameters passed from the higher level. The corresponding notion of a parameterized option is yet to be proposed but it is fairly easy to conceptualize. A relativized option is a special case of a parameterized option in which the policies corresponding to various parameter settings are isomorphic to one another and hence may be compactly represented in a relative space. We can similarly relativize a parameterized HAM or a parameterized sub-MDP in the MaxQ framework.

Feudal reinforcement learning (FRL) (Dayan and Hinton, 1993) is an early hierarchal RL framework that considers both spatial and temporal abstraction. FRL assumes a rigid hierarchy, consisting of "super-managers", "managers" and "sub-managers". A manager is aware only of the task it super-manager sets it and in turn can assign some sub-task to its sub-managers that would help it accomplish its task. Each manager is also aware of the state-space only at a granularity sufficient to be able to accomplish the tasks set it. If it needs anything done at a finer level, then it passes the control to one of its sub-managers. There is a significant amount of work for the designer here, including specifying the hierarchy and also the granularity of the state space at various levels of the hierarchy. Dayan and Hinton do not provide any insight as to how to design such a hierarchy or a useful measure of sufficient granularity. The work is more in the nature of a preliminary investigation and is yet to be taken further.

5.10 Summary

Partial and SMDP homomorphisms extend our abstraction framework to hierarchical RL architectures. We introduced relativized options as a way of compactly representing a related family of sub-tasks in a hierarchical setting. Relativized options facilitate speed up in initial learning performance and enable greater transfer of experience and knowledge between related problem instances. Hierarchical decomposition provides a convenient RL framework that supports simultaneous learning at multiple levels of an hierarchy. We illustrated the utility of relativized options and hierarchical decomposition in a simple grid environment. Employing approximate homomorphisms expands the applicability of relativized options to situations that model a family of similar but not isomorphic tasks.

While relativized options are useful theoretical constructs how applicable are they in practice? If we restrict ourselves to exact homomorphic images, we seldom have situations where relativized options can be applied. But with approximate homomorphisms we can hope to profitably employ them in various problem settings. Relativized options can also form the basis for modeling more powerful abstraction schemes. Abstractions that are useful in learning and adapting a range of skills in the course of a learning agents lifetime. In the next chapter, we discuss some such schemes.

CHAPTER 6
OPTION SCHEMAS AND DEIXIS

In this chapter, we discuss two abstraction approaches that have their roots in developmental models of learning and models of human cognition. Some aspects of these representation can be modeled using extension of relativized options.

6.1 Relativized Options as Option Schemas

In the previous chapter, we introduced relativized options as a way of modeling abstraction in a hierarchical framework. Each relativized option employed its own abstract representation independent of the representation used by the other levels in the hierarchy, and can compactly represent a related family of tasks, like the problems of navigating the 5 rooms in Figure 5.2. We now look at another interpretation of relativized options that is related their ability to model family of tasks.

Let us return to the rooms example from Section 5.6. The one relativized option in the example represents the task of gathering the object in a room and exiting the room. The state space of the option MDP is described by the x and y coordinates and a boolean variable indicating the possession of the object in the room. There are 4 actions: N, S, E, and W. Let us consider what it means when we say that the agent is in the abstract state $\langle 3, 7, 0 \rangle$. Depending on the room in which the agent invoked the option, this represents different states in the original MDP. In room 1, this means that the agent is in location $(3, 7)$ in the room without the object. In room 2, this means the agent is in location $(3, 2)$ in the room, again without the object. In room 5, this means that the agent is in location $(7, 7)$, again sans object. Thus, the numbers 3 and 7 in the option MDP mean different things depending on the context in which the option was invoked. They are in reality placeholders or *variables* that are assigned values when the option is invoked. Thus, the state set of the option MDP is described by a set of 10 x-coordinate variables, 10 y-coordinate variables and one boolean feature. Similarly, the action set is described by 4 action variables that get assigned different actions depending on the context. For example, in room 2, abstract action N is assigned action S and abstract action S is assigned action N.

Thus, a relativized option can be viewed as an *option schema* where a skeleton of an option policy is specified in an abstract space. Evans (1967) defines a schema as: "... a characteristic of some population of objects, and consists of a set of *rules* serving as instructions for producing a population prototype (the concept)." An option MDP is the prototype for a family of problems, and the option homomorphism is the rule for constructing the prototype. When the option is invoked, a particular instantiation of the prototype is chosen by binding the appropriate resources to the schema.

Relativized options can be used to model certain *behavioral schemas*. Behavioral schemas are abstract templates for how to respond to a given situation. When an agent invokes a schema it appropriately allocates various resources and sensory capabilities to make the schema relevant to the specific instance. They provide a very efficient mechanism for generalizing existing skills to new situations.

In the example from Section 5.6, the agent can be trained to navigate in room 1, say, and then in room 2, be instructed to use the same policy as in room 1, with north and south exchanged. As an example closer to the real world, suppose the task of a robot is to assemble different parts of an automobile which require repeated use of a limited set of skills such as tightening a nut, screwing in a bolt, etc. The motor commands required to tighten a nut is the same irrespective of the identity of the nut, but depending on the location and size of the nut, the robot might be required to use different resources, such as different sized wrenches. A robot can be trained to tighten a nut, $nut1$, of some given size. Once the robot learns the skill to a desired level of proficiency, it can be given instructions in terms of the already learned policy, such as "use the same policy as with nut $nut1$ but use a 6.2 head wrench". Instead of learning the policy from scratch, the robot learns to modify the existing policy to accommodate the new situation.

The notion of behavior schemas is a very general one and does not require that we limit our abstract models to homomorphic images of the original system. The schema might take into consideration any arbitrary subset of the local features involved. While it is straightforward to define a hierarchical framework that allows us to represent arbitrary schemas, further analysis of such a framework is difficult and in many cases the practical utility of such arbitrary definitions is not clear. Relativized options that use strict homomorphic images seldom lead to useful schemas in practice. But using approximate homomorphic images allows us to define behavioral schemas in a wide variety of problem settings. In Section 6.3, we demonstrate the utility of a "very" approximate homomorphic image in defining schemas.

6.2 Related Work

The notion of schemas have been widely employed to model a variety of psychological and behavioral phenomena. Bartlett (1932) introduced schemas into psychology. His thesis is that when people recount dreams, they do not do so via a rote memorization of details. They try to "fit" the dream by binding to a set of familiar schemas. Schmidt (1975) proposes a theory of humans motor skills acquisition based on memory and schemas. The initial conditions, response, and consequences are gathered while repeatedly performing some motor commands. Then the skill acquired is stored as a schema specified using the common features of the data gathered across the various trials. Piaget (1952, 1954) proposes a model of cognitive development in children that is based on acquiring sensorimotor schemas. Piaget postulated that at a certain stage of their development children acquire sensorimotor schemas of the world that apply to classes of related instances. When encountering a new situation, the child either "assimilates" it as part of an existing schema or "accommodates" the novel features of the instance by adapting the schemas, acquiring new ones if necessary. Sensorimotor schemas are modeled as being grounded in physical activity and resulting perceptions.

Arbib (1995) proposes schema theory as a basis for distributed computing. This theory builds on the *robot schema* language of Lyons and Arbib (1989). While Arbib demonstrates the viability of the approach in the context of neural computation it is a far more general theory. The basic approach consists of generating schemas, representing interacting sub-problems, which can be combined to produce a solution to the original task. The schemas can be combined sequentially, concurrently or even hierarchically to achieve more complex goals.

In traditional AI literature, one can find notions that are similar to schemas. A production rule can be converted to a *rule schema* (Russel and Norvig, 1995) by incorporating variables in the pre and post conditions, resulting in a skeleton of a rule. When applying this rule, it is instantiated with a particular assignment to the variables. *Knowledge frames* (Russel and Norvig, 1995) can be also be considered as schemas. A frame is a description of an scenario with *slots* for typical objects that are part of the scenario. Assigning objects to these slots corresponds to instantiating a schema.

Drescher (1991) introduces a schema mechanism aimed at emulating the constructive model of cognitive development due to Piaget (1952). Drescher's schema consists of a context, an action and a result. The context and result are abstract representations, specified by assignments to a subset of the features describing the problem. As with Arbib (1995), Drescher also provides a mechanism for generating hierarchical schemas. Composite schemas can be formed by concatenating simple schemas to achieve a particular result. Once a composite schema is formed it may be used in constructing other schemas. But representing stochasticity and probabilistic outcomes in this setting is difficult.

One framework that can accommodate stochasticity and dynamics is the control basis framework (Huber and Grupen, 1999). This framework is built on a set of controllers formed by instantiating abstract control schemas, specified as artificial potentials, with a set of sensors and effectors. The control basis approach is described in more detail in Section 6.7. We also outline one approach to using relativized options in deriving hierarchical option schemas in the control basis framework.

6.3 Choosing Transformations

Given an SMDP, defining a relativized option requires the use of extensive prior knowledge, namely the transition and reward structure of the entire SMDP, at the very least the parts of the SMDP spanned by the desired domain of the option, the parameters of the option SMDP, and the homomorphism that maps the original state and actions onto the option SMDP. It is not often that all this knowledge is available a priori. Even when such knowledge is available, finding the correct option homomorphism is in general NP-hard.

In the absence of complete knowledge about the system, we need methods for learning some of the required components of a relativized options given the others. One useful case is when the agent is given the parameters of the option SMDP and is required to choose the right transformations that constitute the option homomorphism, without complete knowledge of the parameters of the original SMDP. We assume that the agent has access to a set of candidate transformations. The agent learns, using online experience, the right transformation to apply depending on the circumstances under which an option is invoked. When combined, the chosen transformations define the option homomorphism.

Such a scenario would often arise in cases where an agent is trained in a small prototypical environment and is required to later act in a complex environment where skills learned earlier are useful. In the example from Section 5.6, the agent may be trained to gather the object in a particular room and then be asked to gather the objects in the different rooms in the environment. The problem of navigating in each of these rooms can now be considered simply that of learning the suitable transformation to apply to the policy learned in the first room. An appropriate set of candidate transformations in this case are reflections and rotations of the room.

Learning the right homomorphism through experience, can also be viewed as on-line minimization without a completely specified model. Recall from the previous chapter, that an option SMDP, $\mathcal{M}_O = \langle S_O, A_O, \Psi_O, P_O, R_O \rangle$, is a homomorphic image of the original MDP and hence is constructed from the reward respecting SSP partition corresponding to the option homomorphism. Each element of Ψ_O is therefore an unique representative of some block of the partition induced by the option homomorphism. Finding the right transformations is then equivalent to identifying the other members of each of the blocks. Since the agent is restricted to a limited set of transformations, the search for a reward respecting SSP partition of the original SMDP is limited to some fixed family of partitions of Ψ_O.

The interpretation that is intuitively more appealing is the one afforded by viewing relativized options as option schemas. Under this interpretation, the schema, i.e., the option SMDP and the policy, is assumed to be given. The problem is then one of choosing the right bindings to the abstract states and actions from a set of possible bindings. This interpretation yields a better motivation for studying this particular formulation involving insufficient prior knowledge. This is a natural interpretation of our approach to choosing homomorphisms and suggests many problem domains in which this approach may be used.

6.3.1 A Bayesian Algorithm

Formally the problem maybe stated as: Given a set of candidate transformations $\mathcal{H}$ and the option MDP $\mathcal{M}_O = \langle S_O, A_O, \Psi_O, P_O, R_O \rangle$, how do we choose the right transformation to employ at each invocation of the option? We assume that $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$ is the MDP describing the actual task, and that P and R are not known.[1] Let $\psi(s)$ be a function of the current state s that captures the features necessary to distinguish the particular context in which the option is invoked. In the example in Figure 5.2, the room number is sufficient, while in an object-based environment some property of the target object, say color, might suffice. Often in practice, $\psi(s)$ is a simple function of s like a projection onto a subset of features, as in the rooms example. The features, $\psi(s)$, can be thought of as distinguishing various instances of the family of sub-problems represented by the relativized option. The problem of choosing the right transformation is formulated as a family of Bayesian parameter estimation problems, one for each possible value of $\psi(s)$.

There is one parameter, θ, that can take a finite number of values from $\mathcal{H}$. Let $p(h, \psi(s))$ denote the prior probability that $\theta = h$, i.e., the prior probability that h is the correct transformation to apply in the sub-problem represented by $\psi(s)$. The set of samples used for computing the posterior distribution is the sequence of transitions, $\langle s_1, a_1, s_2, a_2, \cdots \rangle$, observed when the option is executing. Note that the probability of observing a transition from s_i to s_{i+1} under a_i for any i, is independent of the other transitions in the sequence. Recursive Bayes learning is used to update the posterior probabilities incrementally.

Let $p_n(h, \psi(s))$ be the posterior probability that h is the correct transformation to apply in the sub-problem represented by $\psi(s)$ after n time steps from when the option was invoked. Initialize $p_0(h, \psi(s)) = p(h, \psi(s))$ for all h and $\psi(s)$. Let $E_n =$

[1]We describe the algorithm in terms of MDPs for simplicity of exposition. The ideas extend to SMDPs naturally.

$\langle s_n, a_n, s_{n+1}\rangle$ be the transition observed after n time steps of option execution. The posteriors for all $h = \langle f, \{g_s | s \in S\}\rangle$ are updated as follows:

$$p_n(h, \psi(s)) = \frac{Pr(E_n | h, \psi(s)) p_{n-1}(h, \psi(s))}{\mathcal{N}}, \tag{6.1}$$

where $Pr(E_n|h, \psi(s)) = P_O(f(s_n), g_{s_n}(a_n), f(s_{n+1}))$ is the probability of observing the h-projection of transition E_n in the option MDP and $\mathcal{N} = \sum_{h' \in \mathcal{H}} P_O((f'(s_n), g'_{s_n}(a), f'(s_{n+1}))p_{n-1}(h', \psi(s))$ is a normalizing factor. When an option is executing, at time step n, $\widehat{h} = \arg\max_h p_n(h, \psi(s))$ is used to project the state to the option MDP and lift the action to the original MDP. After experiencing a transition, the posteriors of all the transformations in $\mathcal{H}$ are updated using equation 6.1.

The term $Pr(E_n|h, \psi(s))$ is a measure of how likely the transition E_n is in the option MDP assuming that the agent is in the sub-problem represented by $\psi(s)$ and h is used to project the transition onto the option MDP. Accumulating this "likelihood" over a sufficiently long sequence of transitions gives the best possible measure of how correct a transformation is for a given sub-problem. $\mathcal{N}$ is a normalizing factor that prevents the values of p_n from decaying to very small numbers with large n. Without normalization, p_n's are likely to decay to zero since the probability of observing any long sequence of transitions is very low even under the correct transformations.

6.3.2 Experimental Illustration

We tested the algorithm on the gridworld in Figure 5.2. The agent, referred to as the Bayesian agent, has one *get-object-and-exit-room* relativized option defined in the option MDP in Figure 5.2(b). $\mathcal{H}$ consists of all possible combinations of reflections about the x and y axes and rotations through integral multiples of 90 degrees. There are only 8 unique transformations in $\mathcal{H}$. Since the rooms delineate the sub-tasks from one another, $\psi(s)$ is set to *room* . For each of the rooms in the world, there is one transformation in $\mathcal{H}$ that is the desired one. This is a contrived example chosen to illustrate the algorithm and reduction in problem size is possible in this domain by more informed representation schemes. We will discuss, briefly, the relation of such schemes to relativized options later in this section. As with earlier experiments reported in Chapter 5, the agent employs hierarchical SMDP Q-learning with ϵ-greedy exploration, with $\epsilon = 0.1$. The learning rate is set at 0.01 and the discount factor, γ, at 0.9. The root task description is the same as in Section 5.6. The priors in each of the rooms were initialized to a uniform distribution with $p_0(h, \psi(s)) = 0.125$ for all $h \in \mathcal{H}$ and $\psi(s)$. The trials were terminated either on completion of the task or after 3000 steps. The results shown in Figure 6.1 are averaged over 100 independent runs.

Recall from Section 5.6, that the probability that an action produces a movement in a direction other than the desired one, is called the *slip* of the environment. The greater the slip, the harder the problem, since the effects of the actions are more stochastic. As shown in Figure 6.1 the agent rapidly learned to apply the right transformation in each room under different levels of stochasticity. Compare this performance to the agent learning with primitive actions alone (Figure 5.4(a)). The primitive action agent didn't start improving its performance until after 30,000 iterations and hence is not employed in further experiments. Figure 6.1 compares the performance of the Bayesian agent with an agent that knew the right transformations apriori. As is illustrated in the figure, the difference in performance is not significant.[2] In this particular task our transformation-choosing algorithm manages to identify the correct transformations without much loss in performance since there is nothing catastrophic in the environment and the agent is able to recover quickly from wrong initial choices.

Figure 6.2 shows how the various posteriors evolve during a typical run. The graph plots the posteriors in room 5 for the transformations composed of pure rotations or

[2]All the significance tests were *two sample t-tests*, with a p-value of 0.01, on the distributions of the learning times under the two algorithms.

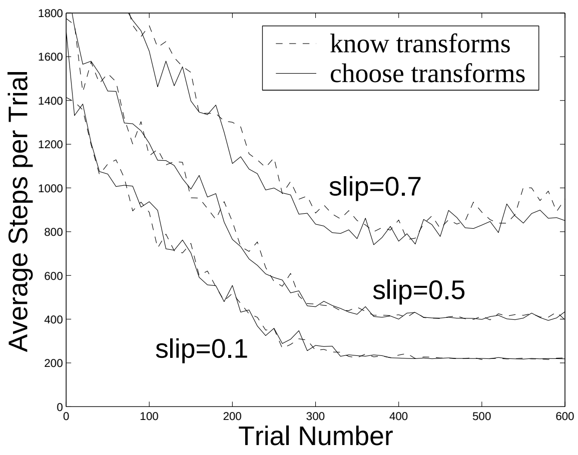


Figure 6.1. Comparison of initial performance of agents with and without knowledge of the appropriate partial homomorphisms on the task shown in Figure 5.2 with various levels of stochasticity.

reflections followed by rotations. The pure reflections were quickly discarded in this case. As is evident, by iteration 10 the posteriors of the incorrect transformations have decayed to 0. The posterior for transform 5, a rotation through 90 degrees, converges to 1.0.

6.4 Handling Approximate Equivalence

The task in Figure 5.2(a) exhibits perfect symmetric equivalence. We return to our modified task shown in Figure 5.7, reproduced as Figure 6.3. Here the different rooms have differently shaped obstacles. As in the earlier experiments, the agent can employ the same features for the option MDP described in Figure 5.2(b) and learn the option policy online. Since the differences between the various rooms are ignored, there is a slight loss in asymptotic performance. But as discussed earlier, this loss can be bounded and reasonable performance can be obtained in this example. In general, the quality of performance in a particular task would depend on the suitability of the approximation employed and the corresponding error bounds.

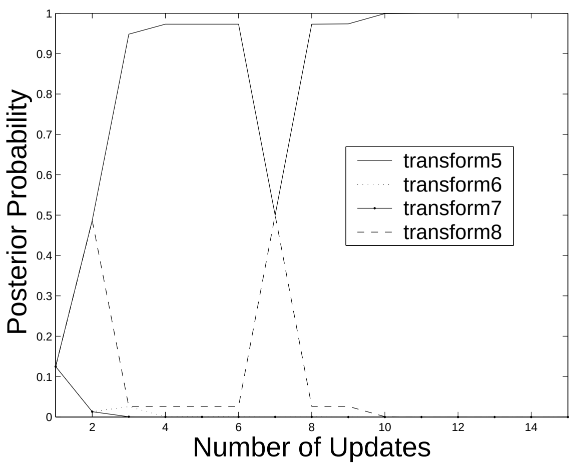


Figure 6.2. Typical evolution of posteriors for a subset of transforms in Room 5 in Figure 5.2, with a slip of 0.1.

The method developed in the previous section for choosing the correct transformations cannot be applied with approximate homomorphisms. In some cases even the correct transformation causes a state transition the agent just experienced to project to an impossible transition in the image MDP, i.e., one with a P_O value of 0. For example, consider moving south from the state marked A in Figure 5.7. This is a valid transition in this room. But when projected onto the option MDP using a rotation through 180 degrees, the correct transformation for this room, the result is an invalid transition. Thus, the posterior probability of the correct transformation might be set to zero, and once the posterior reaches 0, it stays there regardless of the positive evidence that might accumulate later.

To overcome this problem a heuristic may be employed to update Equation 6.1 using a lower bound for P_O values. A weight is computed for each of the transformations using:

$$w_n(h, \psi(s)) = \frac{\overline{P_O}((f(s), g_s(a), f(s')) \cdot w_{n-1}(h, \psi(s))}{\mathcal{N}} \tag{6.2}$$

where $\overline{P_O}(s, a, s') = \max(\nu, P_O(s, a, s'))$, and $\mathcal{N} = \sum_{h' \in \mathcal{H}} \overline{P_O}((f'(s), g'_s(a), f'(s'))$ $w_{n-1}(h', \psi(s))$ is the normalizing factor. This weight serves the role of the posterior probability. Thus, even if the projected transition has a probability of 0 in the option MDP, a value of ν is used for the update. Initialize $w_0(h, \psi(s)) = p(h, \psi(s))$ for all h and $\psi(s)$. The weight $w_n(h, \psi(s))$ is a measure of the likelihood of h being the right transformation in $\psi(s)$ after n transitions, and this weight is used instead of the posterior probability.

We compared the performance of this heuristic on the gridworld in Figure 6.3 to a relativized agent that knew the right transformations for each room. The same parameter settings as described in the previous section were used and the value of ν was set to 0.01. As shown in Figure 6.4 the agent rapidly learns the correct transformation to apply in each room. As was the case earlier, there is no significant difference in performance compared to an agent that already knows the correct transformations. Figure 6.5 shows the evolution of the weights in room 5 during a typical run. Note that the weights decay more slowly than the posterior probabilities do in Figure 6.2.

We presently adopt a "winner take all" transformation selection mechanism, picking the transformation with the largest weight at each time instant. In some situations it is profitable to use the weights as a probability distribution over transformations and select the transformation to use in a given sub-problem by sampling from this distribution. A desirable property of the update rule Equation 6.2 is that it allows the correct transformation's weight to converge to 1 if there is sufficient positive evidence. Hence this heuristic is well suited if one wishes to employ the weights as a probability distribution.

6.5 Experiments in a Complex Game Environment

The experiments reported earlier are more in the nature of a proof of concept. The domains are very simple gridworlds and served to illustrate the concepts developed thus far. Now these ideas are applied to a complex game environment inspired by *Pengi* (Agre, 1988). Pengi is a learning agent that learns to play the video game

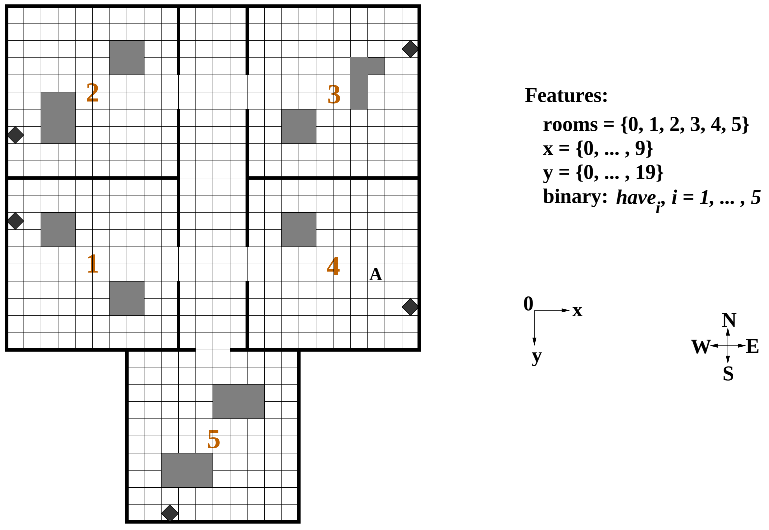

2
3
1
4
A
5
Features:
rooms = {0, 1, 2, 3, 4, 5}
x = {0, ... , 9}
y = {0, ... , 19}
binary: $have_i$, $i = 1, \ldots, 5$
0
x
y
N
W
E
S

Figure 6.3. A simple rooms domain with dissimilar rooms. The task is to collect all 5 objects in the environment.

Pengo. The protagonist of Pengo is a penguin whose objective is to survive in a hostile world inhabited by killer bees. Pengi tries to outrun the bees and can also slide around ice cubes that can be used to crush the bees. The earlier example s modified by adding autonomous adversaries. The task of the agent in the game is to avoid hostile robots while gathering certain objects (diamonds) in the environment. It differs from Pengo in that the agent does not have any weapons available to it.

The layout of the game is shown in Figure 6.6. The environment has the usual gridworld dynamics, i.e., 4 actions, each of which might fail with some probability. When an action fails, it results in a movement in one of the four directions with uniform probability. There are 4 rooms in the world connected by a corridor. The goal of the agent is to collect all the 4 diamonds in the world, one in each room, and return to the central corridor. The agent collects a diamond by occupying the same square as the diamond.

Each room also has several autonomous adversaries. The adversaries may be of two types—benign or delayer. If the agent happens to occupy the same square as

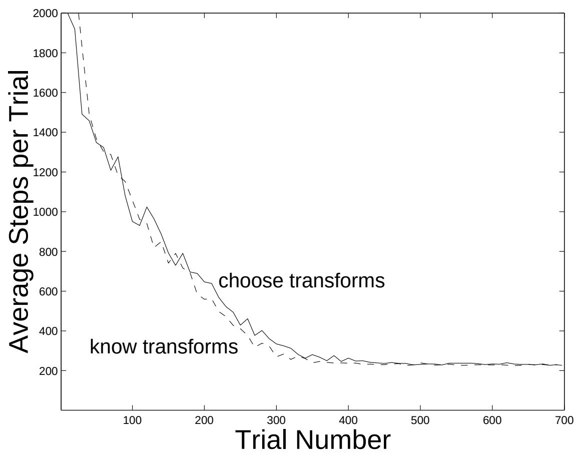


Figure 6.4. Comparison of initial performance of agents with and without knowledge of the appropriate partial homomorphisms on the task shown in Figure 5.7 with a slip of 0.1.

the delayer it is *captured* and is prevented from moving for a random number of time steps determined by a geometric distribution with parameter *hold*. Thus, at each time instant the delayer might release the agent (move away) with probability $(1.0 - hold)$. When not occupying the same square, the delayer pursues the agent with probability *chase*. The benign robots execute random walks in the rooms and act as mobile obstacles. None of the adversaries leave the rooms to which they are assigned. Thus, the agent can escape the delayer in a particular room by exiting to the corridor.

The complete state of the game is described by (1) the position of the agent—the number of the room in which it currently resides (the corridor being 0), and the x and y coordinates in the room; (2) the position of each of the adversary—the number of the room to which they are assigned and the x and y c-ordinates in the room; and (3) boolean variables $have_i$, $i = 1, \ldots, 4$, indicating possession of the diamond in room i. The agent is not aware of the identity of the delayer in each room.

Each room is a 20 by 20 grid, with the origin at the top left corner. The shaded squares in Figure 6.6 are obstacles. The delayers are shown in black and the benign ones are shaded. Note that room 2 does not have a delayer. Each adversary occupies

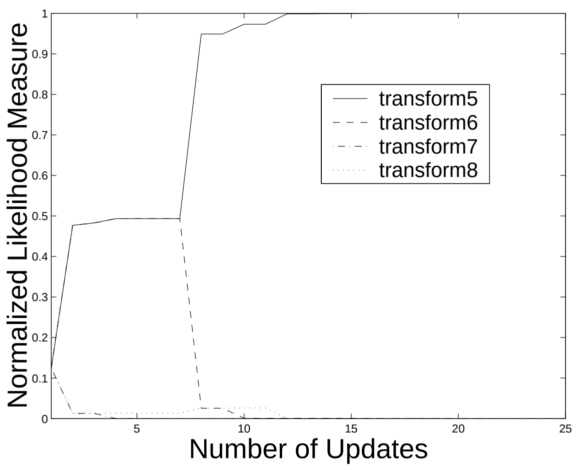


Figure 6.5. Typical evolution of weights for a subset of transforms in Room 5 in Figure 5.7, with a slip of 0.1.

Room	slip	hold	chase
0	0.1	-	-
1	0.15	0.6	0.6
2	0.15	-	-
3	0.1	0.5	0.9
4	0.2	0.4	0.5
option	0.1	0.8	0.5

Table 6.1. Parameter settings in various rooms in Figure 6.6.

one cell in the room. The corridor is 41 cells long. The other parameters in each of the rooms is shown in Table 6.1. The total number of states in this problem is approximately 2.5×10^{56}.

Hierarchical decomposition

In the room example we saw earlier the reduction achieved was due to geometric symmetries such as rotations and reflections. While these symmetries exist in this domain also, we have another source of symmetry that arises dues to the presence of the various adversaries. Therefore apart from considering the usual geometric transformations, we also need to consider permutations of the features corresponding

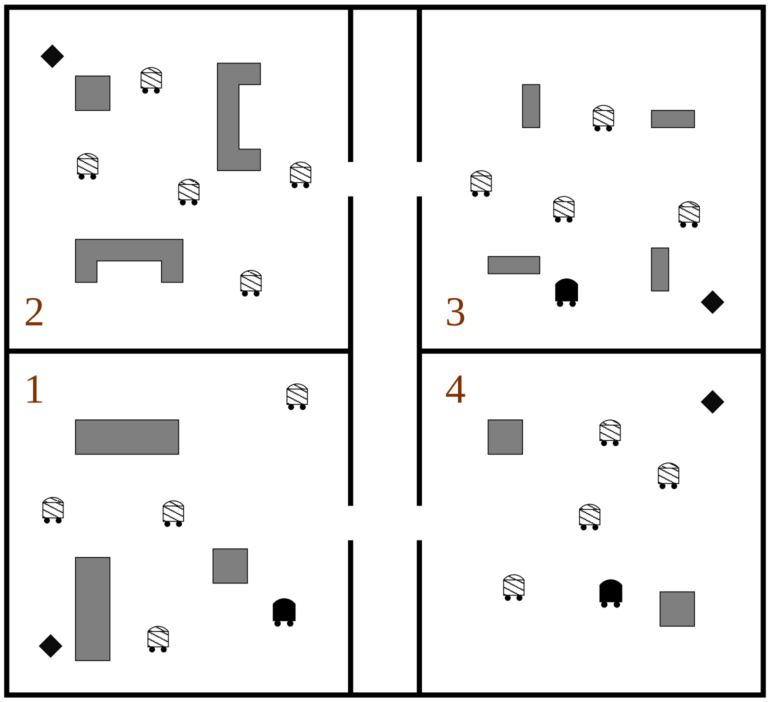


Figure 6.6. A game domain with interacting adversaries and stochastic actions. The task is to collect all 4 objects, the black diamonds, in the environment. The adversaries are of two types—benign (shaded) and delayers (black). See text for more explanation.

to the adversaries. Since a delayer has a very different dynamics from a benign adversary, the permutations should take a delayer's features to another delayer's features. When forming a reduced image of the problem, we want the delayers in each of the rooms to project onto the same adversary in the image.

Our agent has access to one *get-object-and-leave-room* relativized option. The option MDP (Figure 6.7) is a symmetrical room with just one adversary—a delayer with fixed *chase* and *hold* parameters. The features describing the state space of the option MDP are the x and y coordinates, relative to the room, of the agent and of the adversary, and a boolean variable indicating possession of the diamond. None of the rooms in the game match the option MDP exactly and no adversary has the same *chase* and *hold* parameters as this delayer. The initiation set for the option is the set of states in the room, including the doorways. The option terminates on entering the hallway, with or without the diamond.

The root task is as described as follows. The state set of $\mathcal{M}_0$ is described by the number of the room in which the agent currently resides, the various $have_i$ features and if the agent is in the central corridor, then the x and y coordinates of the agent. The admissible actions are the primitive actions in the corridor and the relativized option in the room doorways. The transition and reward functions are those induced by the original task and the option policy. The initiation set is the set of states in the corridor with all $have_i$ features set to FALSE. The termination condition is 1 for states in the corridor with all $have_i$ features set to TRUE. It is 0 elsewhere. This ignores the positions of the adversaries completely, since their positions are not relevant to the root task, given the *get-object-and-leave-room* option.

Note that the option MDP employed here does not conform to either formulation of an approximate homomorphic image we developed in Chapter 3. Both formulations require prior knowledge of the transition dynamics of the MDP, or good estimates of the transition probabilities. In addition, estimates of the visitation frequencies can more accurately reflect the applicability of the option. Usually, when acquiring option schemas access to just one instance of a family of problems is available. The problem designers often have access to idealized models of the problem and can train the agent only in such situations. The option MDP in this experiment was chosen to model such a scenario. One way to overcome this problem is to allow the agent to constantly update the option policy to adapt for situations not encountered in the initial training. Piaget (1952) calls such adaptations of a schema as *accommodation.* However, the results of Theorem 8 can still be used to bound the maximum loss in this setting, provided the parameters of the original MDP are known.

The relativized agent, i.e., the agent that uses the relativized option, has access to a set, $\mathcal{H}$, of 40 transformations consisting of combinations of various spatial transfor-

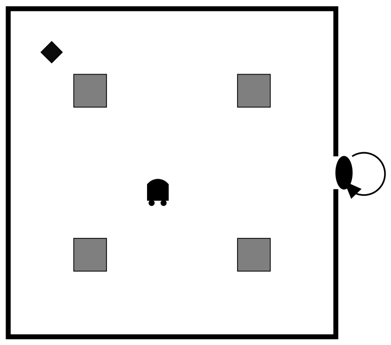


Figure 6.7. The option MDP corresponding to the sub-task *get-object-and-leave-room* for the domain in Figure 6.6. There is just one delayer in the option MDP. The state is described by the x and y coordinates of the agent and the delayer and a boolean feature indicating possession of the diamond.

mations and projections. The spatial transformations consists of various reflections, rotations and their compositions. The reflections are about the x and y axes and about the $x = y$ and $x = -y$ lines. The rotations are through multiples of 90 degrees. Compositions of these change the orientation of the rooms to which they are applied. There are a total of 8 unique spatial transformations. To identify the delayer in each room, the transformation takes the form of a projection, projecting the x and y coordinates of the adversary under consideration onto the option MDP under suitable spatial transformations. There are as many projections as there are adversaries in the room, leading to a maximum of 5 projections. Thus, a total of 40 transformations are available to the agent.

Learning Algorithm

The performance of an agent using one relativized option and 40 candidate transformations s compared with an agent using 4 regular sub-goal options, one for each room. Both the agents employ hierarchical SMDP Q-learning to simultaneously learn the higher level policy and the option policies. The learning rate is set at 0.05 for the higher level and 0.1 for the option. The discount factor γ is set to 0.9 and ϵ to 0.1

for both the higher level and the option. The learning trials are terminated either on completion of the task or after 6000 steps.

The prior weights are initialized for each of the transformations in each of the rooms to a uniform distribution $w_0(h, \psi(s)) = 0.025$ for all $h \in \mathcal{H}$ and for all $\psi(s)$. The agent uses equation 6.2 to update the weights for the transformations, with ν set to 0.01. Since none of the rooms match the option MDP, keeping the option policy fixed leads to very poor performance. So, as mentioned earlier, we allowed the agent to continually modify the option's policy while learning the correct transformations.

Results

The experiments are averaged over 10 independent runs. As shown in Figure 6.8, the agent using the heuristic shows rapid improvement in performance initially. This supports the contention that it is easier to learn the correct transformations than to learn the policies from scratch. As expected, the asymptotic performance of the regular agent is better than the relativized agent. The heuristic could not be compared against an agent that already knows the correct transformations since there are no correct transformation in some of the cases. Figure 6.9 shows the evolution of weights in room 4 during a typical run. The weights have not converged to their final values after 600 updates, but transformation 12, the correct transformation in this case, has the largest weight and is picked consistently. After about thousand updates the weight for transformation 12 reach nearly 1 and stay there. Figure 6.10 shows the evolution of weights in room 2 during a typical run. The weights oscillate a lot during the runs, since none of the transforms are entirely correct in this room. In this particular run, the agent converges to transformation 5 after about 1000 updates, but that is not always the case. But the agent can solve the sub-task in room 2 as long as it correctly identifies the orientation and employs any of transformations 1 through 5.

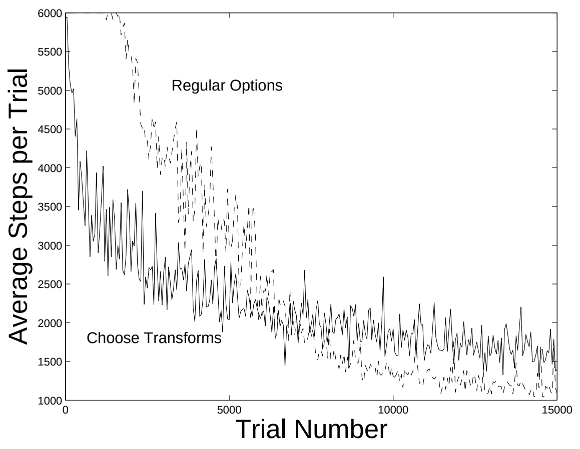


Figure 6.8. Comparison of the performance of an agent with 4 regular options and an agent using a relativized option and no knowledge of the correct transformation on the task shown in Figure 6.6(a). The option MDP employed by the relativized agent is shown in Figure 6.6(b).

Discussion

For some problems it is possible to choose representation schemes to implicitly perform the required transformation depending on the sub-task. While employing such representations largely simplifies the solution of a problem, they are frequently very difficult to design. Our work is a first step toward systematizing the transformations needed to map similar sub-tasks onto each other in the absence of versatile sensory mechanisms. The concepts developed here will also serve as stepping stones to designing sophisticated representation schemes. Examples of such schemes include ego-centric and deictic representations (Agre, 1988).

Agre (1988) used the Pengo environment to demonstrate the utility of deictic representations. Deictic representation is a form of indexical representation and consists of sensing the world through a set of pointers. In the Pengo domain, the agent used pointers like *bee-attacking-me*, *icecube-next-to-me* etc., to model the world. In our domain, the option MDP models the behavior of a delayer, i.e., that of chasing the agent and capturing it. Finding the right projection to the option MDP can be thought of

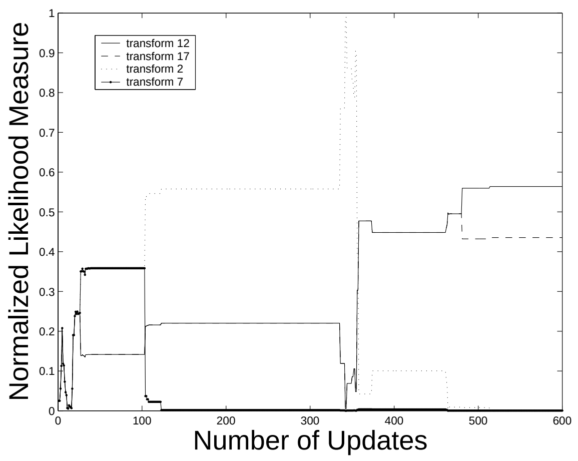


Figure 6.9. Typical evolution of weights for a subset of transformations in Room 4 in Figure 6.6(a), with a slip of 0.1.

as trying to place a pointer on the *adversary-chasing-me*. Thus, relativized options along with a mechanism for choosing the right transformations can be used to model certain aspects of deictic representations.

6.6 Deictic Representation

A wide variety of complex tasks are naturally modeled as collections of objects, their properties and their interactions. Objects might range from simple blocks and tools to clouds and adversaries. One approach to describing such environments is by a set of predicates, one for each property of a object that is relevant. Operators act on one or more objects, changing certain properties of the objects. In an MDP framework a naive way to represent such an environment is to model it as a structured MDP, with the state set described by a huge array of features, one for each predicate needed in a classical representation, and the actions affecting only a subset of such features. In other words, in an object-based factored MDP, $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$, the state set S is described by $mN + B$ features, where m is the number of objects in the world, N the number of predicates needed to describe each object and B is the number of

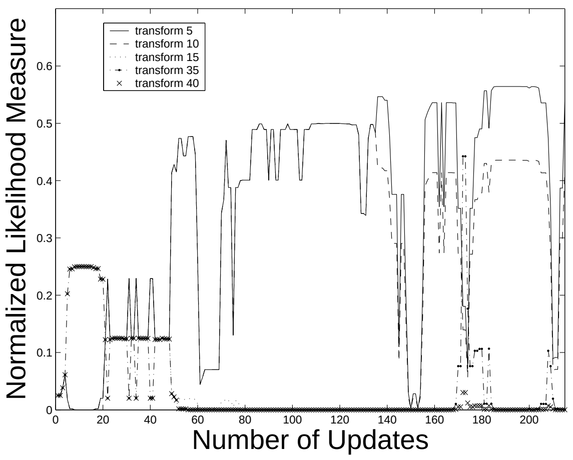


Figure 6.10. Typical evolution of weights for a subset of transformations in Room 2 in Figure 6.6, with a slip of 0.1.

features needed to represent other aspects of the environment. Each action $a \in A$ acts only on one object and affects only a subset of these features. This is a special case of a factored MDP, one in which there is a lot of independence among features and effects of actions. The 2-TBNs representing such MDPs are largely decoupled and hence amenable to various, possibly partial, homomorphic abstractions.

The above approach to modeling object-based environments invariably leads to a plethora of features and associated problems. Researchers have employed various ideas to make operating in such environments feasible (e.g. Boutilier et al., 2001; Koller and Pfeffer, 1998; Getoor et al., 2001). Deictic representation (Agre, 1988; Agre and Chapman, 1987; Ballard et al., 1996), based on pointing, is one such paradigm. The environment is sensed via multiple pointers and actions are specified with respect to these pointers. Consider the example in Figure 6.11. The task in Figure 6.11(a) is to place block B on block A and in Figure 6.11(b) is to place block X on block Z. If we have a pointer indicating the lower block (shown as a $+$) and a pointer indicating the upper block (shown as a $\times$), then both the problems reduce to that of placing the block pointed to by $\times$ on the block pointed to by $+$.

In this example, the effective state of the system is represented by the attributes of the blocks on which the pointers are placed, attributes like color and location. Thus, in effect the pointers project the complete state of the system described by the attribute of all the blocks in the domain, to an abstract space described the attributes of two blocks. The blocks whose attributes are projected are determined by the two pointers. Actions in this abstract space are of the form "move the $+$ block to the top of the $\times$ block", "move the $\times$ block to the top of the table", etc. Depending on the actual location of the pointers, the blocks moved vary.

Deictic pointers might be simple physical locators like those above or maybe arbitrarily complex. Agre and Chapman (1987) employ pointers that need substantial pre-processing and domain knowledge, that let the agent precisely locate important components of the system. While solving the arcade game Pengo, their agent Pengi employs complex pointers such as *bee-attacking-me*, *ice-cube-next-to-me*, etc. The actions of the agent are then defined with respect to these pointers, for e.g. *push ice-cube-next-to-me toward bee-attacking-me*. This enables them to reduce an intractable problem to a more manageable size and results in a satisfactory player.

In general, deictic representations can be used in rich environments with incredible amounts of detail. Deixis helps limit the attention of the agent to a few features in the environment that are relevant to the task at hand. It is also useful in systems where there are physical limitations on the sensory capabilities and perforce they have to use some form of attentional mechanism (Minut and Mahadevan, 2001).

6.6.1 Modeling Aspects of Deixis with Relativized Options

As mentioned briefly in Section 6.5, some transformations applied to the state space of a MDP to project it onto an option MDP can be viewed as a form of deictic representation. Looking at it from another perspective, some set of deictic pointers, together with their possible configurations, specify a set of candidate transformations,

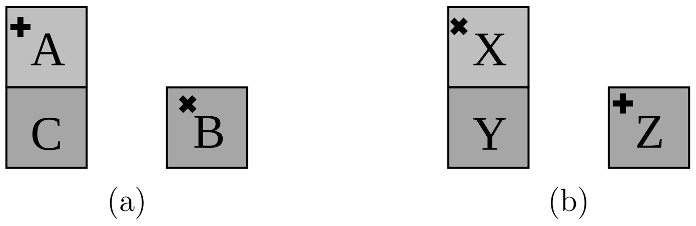


Figure 6.11. Deixis in a simple blocks world domain: The task in (a) is to place block B on A, while in (b) is to place X on Z. The two tasks reduce to that of placing block pointed to by $\times$ over block pointed to by $+$.

or bindings, for an option schema. The agent learns to place the pointers in specific configurations to effect the correct bindings to the schema. We call such option schema together with the set of pointers a *deictic option schema*. Formally a deictic option schema is defined as follows:

Definition: A *deictic option schema* of a factored SMDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$ is the tuple $\langle K, \mathcal{D}, O\rangle$, where $O = \langle h, \mathcal{M}_O, \mathcal{I}, \beta\rangle$, is a relativized option in $\mathcal{M}$, K is the number of deictic pointers available and $\mathcal{D} = \{D_1, D_2, \cdots, D_K\}$ is the set of admissible configurations of the deictic pointers. For all i, $D_i \subseteq 2^{\{1,\cdots,M\}}$ is the collection of all possible subsets of indices of the features that pointer i can project onto the schema. M is the number of features used to describe S.

The set D_i indicates the set of objects that pointer i can point to in the environment. In the blocks world example in Figure 6.11, this is the set of all blocks. Therefore each element of D_+ or $D_\times$ comprises of the indices of the features corresponding to the attributes of any one block in the environment. Thus, if features 1 and 2 are the color and position of block A, and features 3 and 4 are the color and position of block B, and so on, the elements of D_i are of the form $\{2j-1, 2j\}$ for $j = 1, \cdots m$, where m is the number of blocks in the environment.

Recall from Chapter 5 that the option MDP $\mathcal{M}_O$ is a partial homomorphic image of the original SMDP with a suitable reward function. Let $\mathcal{H}$ denote the set of candidate transformations from among which the option homomorphism, h, should be constructed. Each member of $\mathcal{H}$ has a state transformation of the form $\prod_{i=1}^{K} \rho_{J_i}$, where $J_i \in D_i$ for all i and ρ_J is the projection of S onto the subset of features indexed by J. If h is known a priori then the pointer configurations can be chosen appropriately while learning. In the absence of prior knowledge the Bayesian algorithm developed in Section 6.3.1 can be used to determine the correct bindings to the schema from among the possible pointer configurations. But, the algorithm is not entirely suitable for deictic option schemas for the following reason.

The algorithm assumes that the candidate transformations are not structured and maintains a monolithic weight vector, $w_n(\cdot, \cdot)$. In the case of deictic option schemas the transformations are structured and it is advantageous to maintain a "factored" weight vector, $w_n(\cdot, \cdot) = \langle w_n^1(\cdot, \cdot), w_n^2(\cdot, \cdot), \cdots \rangle$. Ideally each component of the weight should be the likelihood of the corresponding pointer being in the right configuration. But usually there is a certain degree of dependence among the pointers and the correct configuration of one pointer depends on the configuration of other pointers.

Therefore, three cases need to be considered. Assume that there are only two pointers, i and j, for the following discussion, but the concepts generalize to arbitrary number of pointers.

1. *Independent pointers:* For every $J_i \in D_i$, ρ_{J_i} satisfy the homomorphism condition on transition probabilities given by equation 4.2. Then, the right assignment for pointer i is independent of the other pointers and there is one component of the weight vector corresponding to pointer i and the updates for this components depends only on the features indexed by some J_i.

2. *Mutually dependent pointers:* For each $J_i \in D_i$ and $J_j \in D_j$, $\rho_{J_i} \times \rho_{J_j}$ satisfies equation 4.2. But ρ_{J_i} and ρ_{J_j} do not satisfy equation 4.2 for some J_i and J_j. Thus, they cannot be treated separately and the composite projections given by their cross-products has to be considered. There is one component of the

weight vector that corresponds to this cross-product projection. The update for this component will depend on the features indexed by some J_i and J_j.

3. *Dependent pointer:* For each $J_i \in D_i$ and $J_j \in D_j$, $\rho_{J_i} \times \rho_{J_j}$ satisfies equation 4.2, as does ρJ_i. But ρ_{J_j} does not satisfy equation 4.2 for at least some value of J_j. This means pointer i is an independent pointer, while j is a dependent one. There is a separate component of the weight vector that corresponds to pointer j, but whose update depends on the features indexed by both J_i and J_j.

The weight vector is chosen such that there is one component for each independent pointer, one for each dependent pointer and one for each set of mutually dependent pointers. Let the resulting number of components be L. A modified version of the update rule Equation 6.2 is used to update each component l of the weight independently of the updates for the other components:

$$w_n^l(h^i, \psi(s)) = \frac{\overline{P_O^l}((f^i(s), g_s^i(a), f^i(s')) \cdot w_{n-1}^l(h^i, \psi(s))}{\mathcal{K}} \tag{6.3}$$

where $\overline{P_O^l}(s, a, s') = \max\left(\nu, P_O^l(s, a, s')\right)$ and $\mathcal{K} = \sum_{h'^i \in \mathcal{H}} \overline{P_O^l}(f'^i(s), g_s'^i(a), f'^i(s'))$ $w_{n-1}(h'^i, \psi(s))$ is the normalizing factor. $P_O^l(s, a, s')$ is a "projection" of $P_O(s, a, s')$ computed as follows. Let J be the set of features that is required in the computation of $w_n^l(h^i, \psi(s))$. This is determined as described above for the various cases. Then $P_O^l(s, a, s') = \prod_{j \in J} \text{Prob}(s_j' | \text{Parents}(s_j', a))$.

6.6.2 Experimental Illustration in a Complex Game Environment

We now apply a deictic option schema to learning in a modified version of the game environment introduced in Section 6.6. The layout of the game is shown in Figure 6.12. The environment has the usual gridworld dynamics as described earlier. Unlike in the previous example there is just one room in the world and the goal of the

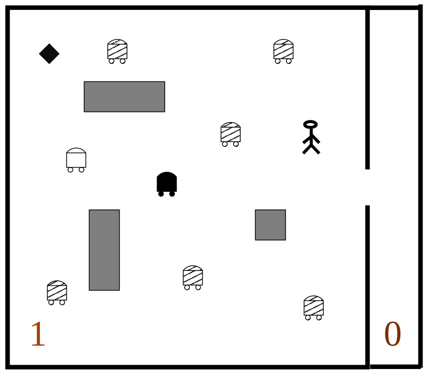


Figure 6.12. A modified game domain with interacting adversaries and stochastic actions. The task is to collect the black diamond. The adversaries are of three types—benign (shaded), retriever (white) and delayers (black). See text for more explanation.

agent is to collect the diamond in the room and exit it. The agent collects a diamond by occupying the same square as the diamond.

The room also has 8 autonomous adversaries. The adversaries may be of three types—benign, delayer or retriever. The behavior of the benign and delayer adversaries are as described earlier. The retriever behaves like the benign adversary till the agent picks up the diamond. Once the agent picks up the diamond, the retriever's behavior switches to that of the delayer. The main difference is that once the retriever occupies the same square as the agent, the diamond is returned to the original position and the retriever reverts to benign behavior. The adversary returns to benign behavior if the agent is also "captured" by the delayer.

The complete state of the world is described by (1) the position of the agent—the number of the room it is currently in (the corridor being 0), and the x and y coordinates in the room; (2) the position of each of the adversaries given by the x and y coordinates in the room; and (3) a boolean variable *have* indicating possession of the diamond. The other parameters of the task are as defined earlier. The agent is not aware of the identity of the delayer or the retriever. The shaded squares in Figure

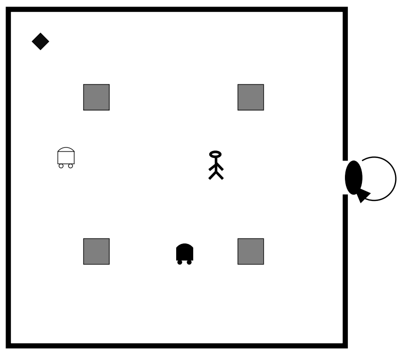


Figure 6.13. The option MDP corresponding to the sub-task *get-object-and-leave-room* for the domain in Figure 6.12. There is just one delayer and one retriever in this image MDP.

6.12 are obstacles. The delayers are shown in black, the retrievers in white and the benign ones shaded.

The option MDP (Figure 6.13) is a symmetrical room with just two adversaries—a delayer and a retriever with fixed *chase* and *hold* parameters. The features describing the state space of the option MDP consists of the x and y coordinates relative to the room of the agent and of the adversaries and a boolean variable indicating possession of the diamond. The rooms in the world does not match the option MDP exactly and no adversary in the world has the same chase and hold parameters as the adversaries here. The root task is the same one described earlier in Section 6.6.

The deictic agent has access to 4 pointers: A *self* pointer that projects the agent's location onto the image MDP, a *have* pointer that projects the have feature onto the image MDP, a *delayer* pointer that projects one of the adversaries onto the delayer in the image MDP and a *retriever* pointer that projects one of the adversaries onto the retriever in the image MDP. The *self* pointer is an independent pointer. The *delayer* pointer is dependent on the self pointer and the *retriever* pointer is dependent on all the other pointers. Note that the *self* and *have* pointers are fixed projections and have very restricted domains. The sets $D_{delayer}$ $D_{retriever}$ are given by the 8 pairs of features describing the adversary coordinates. D_{self} is a singleton consisting the agent's x and y coordinates with respect to the room and D_{have} is also a singleton consisting of the *have* feature.

Traditionally, such pointers are not viewed as deictic pointers, but would form part of the "background" information provided to the agent. But we treat them as special pointers in our formulation. The actions of the agent in the image can then be expressed with respect to the *self* pointer. Note that since the option MDP is an approximate homomorphic image, the homomorphism condition 4.2 is not strictly met by any of the projections. Therefore, in computing the weight updates, the influence of the features not used in the construction of the image MDP are ignored by marginalizing over them.

Experimental Results

The performance of the deictic agent is compared with a relativized agent that employs the same option MDP but chooses from a set $\mathcal{H}$ of 64 monolithic transformations, formed by the cross product of the 8 configurations of the deictic pointers. Both agents employ hierarchical SMDP Q-learning, with the learning rates for the option and the root task set to 0.1. The agents are both trained initially in the option MDP to acquire an approximate initial option policy that achieves the goal some percentage of the trials, but is not optimal. Both agents use ϵ greedy exploration.

On learning trials both agents perform similarly, with the monolithic agent having better initial performance. This is not surprising, if we look at the rate at which the transformation weights converge. Figure 6.14 shows that the monolithic agent identifies the right transformation rapidly just as the deictic agent identifies the *delayer* rapidly (Figure 6.15(a)). But as Figure 6.15(b) shows, identifying the *retriever* takes much longer, and the deictic agent performs poorly till the retriever is correctly identified consistently.

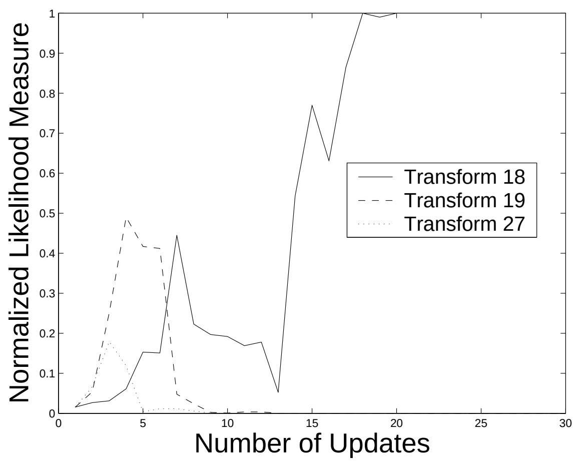
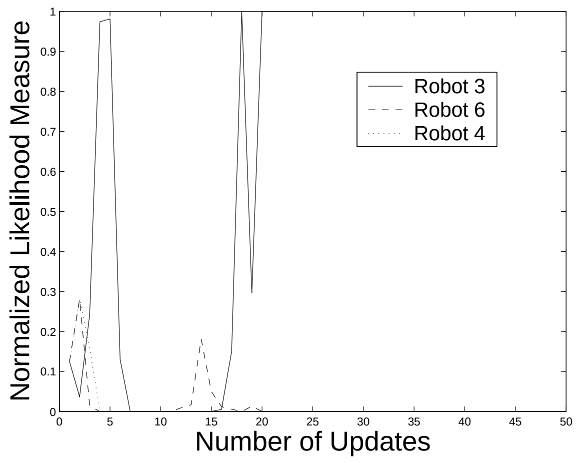


Figure 6.14. Typical evolution of a subset of weights of the monolithic agent on the task shown in Figure 6.12.

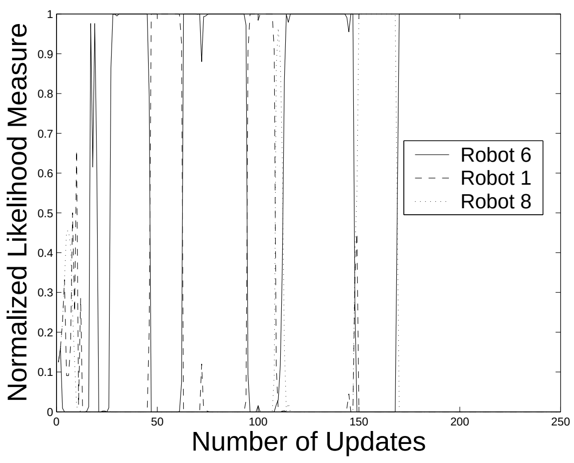
This result is not surprising, since the correct position for the *retriever* depends on position of the *delayer* pointer. Therefore, while the *delayer* is being learned, the weights for the *retriever* receive inconsistent updates and it takes a while for the weights to get back on track. Further, the monolithic agent considers all possible combinations of pointer configurations simultaneously. Therefore, while it takes fewer update steps to converge to the right weights, both agents make comparable number of weight updates, 1300 vs. 1550.

Discussion

The algorithm used above updates the weights for all the transformations after each transition in the world. This is possible since the transformations are assumed to be mathematical operations and the agent could use different transformations to project the same transition onto to the option SMDP. But deictic pointers are often implemented as physical sensors. In such cases, this is equivalent to sensing every adversary in the world before making a move and then sensing them after making the move, to gather the data required for the updates. Since the weights converge



(a)



(b)

Figure 6.15. (a) Typical evolution of a subset of the *delayer* weights of the deictic agent on the task shown in Figure 6.12. (b) Typical evolution of a subset of the *retriever* weights on the same task.

fairly rapidly, compared to the convergence of the policy, the time the agent spends "looking around" would be a fraction of the total learning time.

6.6.3 Perceptual Aliasing with Deictic Representations

The power of deixis arises from its ability to treat many perceptually distinct states in the same fashion, but it is also the chief difficulty in employing deictic representations. Consider the example in Figure 6.16. In both situations depicted in the figure, the pointers return the same information, and hence both situations are considered equivalent. If the task is to stack three different colored blocks on top of each other, we can move $\times$ block on top of $+$ block in Figure 6.16(b), while in the other case, moving $+$ to the top of $\times$ is a better choice. Thus two qualitatively different states are mapped onto the same representation here. This phenomenon is known as *perceptual aliasing* (Whitehead and Ballard, 1991).

One approach to overcome perceptual aliasing is a class of methods known as *Consistent Representation Methods.* These methods split the decision making into

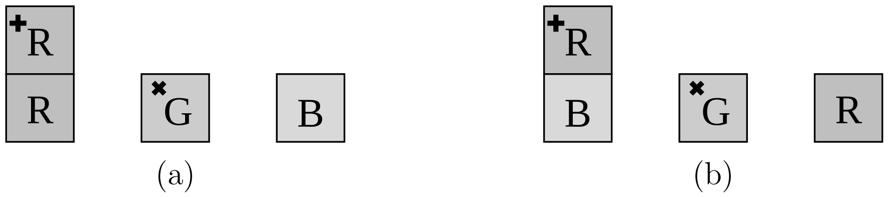


Figure 6.16. Perceptual aliasing in a simple blocks world domain: Both (a) and (b) map to the same representation and are treated as equivalent. But the task is to stack blocks of three different colors and the desired action in the two situations are different.

two phases: in the perceptual phase the agent looks around the environment to find a consistent representation of the underlying state. A consistent representation (Whitehead and Ballard, 1991; Whitehead and Lin, 1995) of a state is one such that all states that map to the representation have the same optimal action-value function. In the overt phase the agent picks an action to apply to the environment based on the current sensory input. Learning takes place in both phases. The *Lion* algorithm (Whitehead and Ballard, 1991) is an example of a consistent representation algorithm. Here Q-learning is used in the overt phase and a simple learning rule based on one step error information is used to train the sensory phase. If the one step error in the Q update rule for a particular configuration is negative then that representation is considered perceptually aliased and is ignored in the future. This simple rule limits the applicability of this algorithm to deterministic settings alone.

If the representation used is a homomorphic image then it is a consistent representation, from Theorem 1. By restricting the definition of deictic option schema to employ partial homomorphic images as option MDPs, it is guaranteed that a consistent representation is always employed. In the absence of knowledge of the option homomorphism, finding the right transformation to employ constitutes the search for a consistent representation and we employ Bayes learning in this phase. As with the Lion algorithm, a form of Q-learning is used in the overt phase.

Another approach to dealing with perceptual aliasing is to use memory based methods (McCallum, 1995; Finney et al., 2002). The basic idea is to remember sufficient history information to be able to distinguish states that are perceptually aliased. These methods use statistical tests to determine how much memory is required and to selectively determine which experiences to retain. These can also be viewed as consistent representation methods, since they aim to remember sufficient data so that the current memory and sensory input together constitute a consistent representation.

6.6.4 Related Work

Deixis originates from the Greek word *deiknynai* which means to show or to point out. It is employed by linguists (Jarvella and Klein, 1982) to denote the pointing function of certain words, like *here* and *that*, whose meaning could change depending on the context. Deictic representations are used in developing models of cognition and visual attention (Land et al., 1998; Ballard et al., 1996; Howarth and Buxton, 1993). Deixis was introduced to the AI community by Agre (1988). Agre and Chapman (1987) used deictic representations to design an agent, Pengi, that plays the arcade game Pengo. Pengi was designed to play the game from the view point of a human player and hence used visuals from a computer screen as input. Agre employed Ullman's visual routines (Ullman, 1984) to extract information and maintain complex deictic pointers such as *bee-attacking-me*. Chapman (1991) later designed Sonja, an instruction taking agent that plays the game amazon, employing deictic commands and representations.

Whitehead and Ballard (1991) were the first to use deictic representations in a RL system, with their *Lion* algorithm. Unfortunately, the method the Lion algorithm employs to determine consistency works only in deterministic environments. McCallum (1995) takes a more direct approach to overcoming perceptual aliasing. He employs deixis to solve a car driving task and models the problem as a partially observable MDP (Kaelbling et al., 1998). He uses a tree structure, known as U-trees, for representing "states" and identifies the necessary distinctions so that the resulting representation is consistent. But his approach is not divided into explicit perceptual and overt phases. There has not been much work on using hierarchical RL and deixis. The only work we are aware of is by Minut and Mahadevan (2001). They develop a selective attention system that searches for a particular object in a room. It operates by identifying the most salient object in the agent's visual field and shifting its visual field to center and focus on that object. They employ an option to identify the most salient object in the current visual field. Though they do not state it thus, this is a "deictic" option, whose effect depends on the current visual field.

Another recent successful application of deixis albeit in a carefully hand crafted fashion is due to Cleary (1997). He solves a robot navigation task by programming a small number of deictic commands and using expert knowledge to position the pointers required by these commands. He demonstrates that with a fairly small number of deictic commands the robot is able to successfully navigate open environments such as the college grounds.

A systematic study on using deictic representations with RL was reported by Finney et al. (2002). They employ a straightforward deictic representation with two pointers on a blocks world task. They use the G-algorithm to represent past information as a tree. They report that their approach does not work well for a variety of reasons. First the tree grows very large rapidly. The deictic commands are defined with respect to the two focus pointers. When long sequences of actions are required with a small number of pointers, it is easy to lose focus. While they try to address this by redesigning the pointers, they do not have much success. One way to alleviate this problem is by adopting a hierarchical approach as we do in this work. If the number of pointers required by each deictic level to maintain focus is not large, we can avoid some of the problems encountered by Finney et al. (2002).

6.6.5 Discussion

While deixis is a powerful paradigm ideally suited for situations that are mainly reactive, it is difficult to employ a purely deictic system to solve complex tasks that require long-range planning. Our hierarchical deictic framework allows us to employ deictic representations in lower levels of the problem to leverage their power and generalization capacities, while at the higher levels we retain global context information in a non-deictic fashion. Mixing such representations allows us to exploit the best of both worlds and to solve tasks that require maintaining long term focus. It is our belief that there is no pure deictic system in nature. While it has been established that humans employ deixis in a variety of settings (Ballard et al., 1996; Land et al., 1998), we certainly maintain some higher level context information. While gathering ingredients for making tea, we might be using deictic pointers for accessing various containers (Land et al., 1998), but we also are continuously aware of the fact that we are making tea.

The various approaches to learning with deictic pointers (Whitehead and Ballard, 1991; Finney et al., 2002) usually employ simple pointers similar to those in Figure 6.11. Agre (1988) uses complex pointers, but hand coded the policy for maintaining the focus of these pointers. For example, a set of rules are used to determine which is the *bee-attacking-me* and the pointer is moved suitably. Our approach falls somewhere in between. We start by defining a set of simple pointers. As learning progresses the agent learns to assign these pointers consistently that some of them take on complex roles. We can then assign semantic labels to these pointers such as *robot-chasing-me*.

It should be noted that a homomorphic image implies a consistent representation but the reverse is not true. The notion of a consistent representation is a more general concept than a homomorphic image and corresponds to optimal-action value equivalence discussed in Chapter 3. By restricting ourselves to homomorphic images we are limiting the class of deictic pointers that we can model. Further work is needed to extend our framework to include richer classes of deictic pointers and to employ memory based methods. Nevertheless in this thesis we have taken the first steps in accommodating deictic representations in a hierarchical decision theoretic framework.

6.7 Applications of Homomorphisms in the Control Basis Framework

Discrete MDPs and SMDPs are useful mathematical constructs that allow us to model a variety of stochastic decision problems. Thus far, we have developed our framework in settings that are idealized representations of the environment. This enabled us to rigorously formulate the concepts of homomorphisms and abstraction. In order to apply our ideas to real problems however, several steps that formalize the domain knowledge are needed.

The problem must first be formulated as an MDP or SMDP. This involves selecting a suitable state and action representation. Second, the problem must be hierarchically structured. Any available knowledge about the various sub-problems of the original task can be exploited in this step. In this section, we explore the applications of our techniques in a domain that is modeled using the control basis framework, a flexible and powerful hierarchical architecture. It is also possible to learn the hierarchical structure, and this is an active research area (Digney, 1998, 1996; McGovern and Barto, 2001; Pickett and Barto, 2002; Hengst, 2002). This hierarchical structure is then be used to create appropriate option schemas.

Finally, a family of transformations for the option schema must be selected. Since this family may consist of all plausible transformations, it is possible to specify it with very little domain knowledge. In navigation tasks, it consists of all reasonable geometric transformations, as in the rooms example. In worlds with objects, it consists of all possible permutations among features corresponding to distinct objects, followed by a suitable projection onto the option SMDP. Once the option schema is

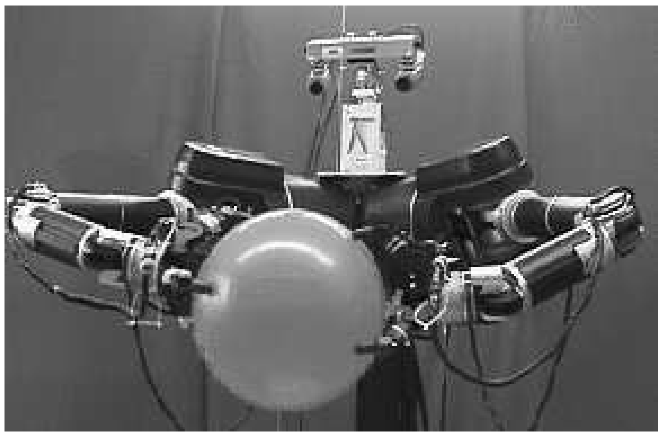


Figure 6.17. The UMass Torso holding a ball in a two handed grasp.

acquired, then the agent can learn to solve the task, using the Bayesian algorithm as described in Section 6.3.

In this section, these issues are explored in the context of a robotics application domain. The platform adopted is the UMass Torso (Platt Jr. et al., 2003, 2004), a partial humanoid robot, with two hands, two arms and a stereo vision head (Figure 6.17). Successful robots must operate in open environments that are not completely controllable. It is almost impossible to model the interaction of the UMass Torso with its environment at a joint and sensor level as an MDP or SMDP. Therefore we turn to a control architecture known as the *control basis framework* proposed by Huber (2000). This control architecture has been used as the basis for learning walking gaits on a quadruped walking platform (Huber and Grupen, 1999), learning multi-fingered grasps (Coelho and Grupen, 1997), designing controllers for whole body grasps (Platt Jr. et al., 2003, 2004), and in controlling ad hoc mobile robot networks (Sweeny, 2003).

6.7.1 Control Basis Framework

The control basis approach is a framework for combining closed loop controllers in a systematic way to accomplish a variety of different objectives. In this approach a controller ${}^{i}\Phi^{\mathcal{S}}_{\mathcal{E}}$ is constructed by associating, or binding, a set of sensors, $\mathcal{S}$, and a set of effectors, $\mathcal{E}$, with an objective function, or artificial potential, ${}^{i}\Phi$. The artificial potential describes the objective i, where i might be a stable leg configuration or a firm grasp on an object. When the controller achieves the minimum potential it is said to have converged (to its objective).

In terms of the terminology developed earlier, the artificial potential that describes the control objective can be viewed as an option schema and the various sensor and effector bindings are the transformations. For example, in the quadruped walking task (Huber, 2000), a schema specifies how to achieve a stable 3 legged stance, and the transformations choose 3 of the 4 legs of the robot to bind to the schema. Huber achieves a turning gait by executing this schema repeatedly with a suitable sequence of transformations.

This framework also allows two or more controllers to execute concurrently in a prioritized manner. Since control actions are derived by descending artificial potentials, a secondary control action will not interfere if it moves the robot along an equipotential line of the primary control potential. This can be achieved if the secondary controller is constrained to operate in the null space of the primary controller. The term *subject to* is used to describe this prioritization and ${}^{i}\Phi^{\mathcal{S}_i}_{\mathcal{E}_i} \triangleleft {}^{j}\Phi^{\mathcal{S}_j}_{\mathcal{E}_j}$ indicates that ${}^{i}\Phi^{\mathcal{S}_i}_{\mathcal{E}_i}$ operates subject to ${}^{j}\Phi^{\mathcal{S}_j}_{\mathcal{E}_j}$.

Closed-loop controllers as defined above are robust to noise and transform a continuous state space into a set of discrete states corresponding to the convergent states of the various controllers. For example, a controller for grasping an object descends an artificial potential, such that the basin of attraction corresponds to a good grasp configuration. When invoked, this controller causes a transition from an arbitrary initial state to a state corresponding to a good grasp. If all the actions are drawn from closed-loop controllers, only a discrete set of states comprising of the equilibrium states of the controller need be considered. Thus the problem can be modeled at the level of the controllers as an MDP with a set of closed-loop controllers as the actions and the states described by the convergence status of the controllers (Huber, 2000; Huber and Grupen, 1999). The transition times of the primitive controllers can be incorporated by modeling the problem as an SMDP. The next step in the design is to specify a hierarchical task decomposition that uses these closed-loop controllers as primitive actions.

6.7.2 Designing Option Schemas

For the purposes of this discussion we focus on tasks based on whole body grasping (Platt Jr. et al., 2003, 2004). The goal is to achieve a certain objective that requires the robot to manipulate objects while using different contact resources to maintain a good grasp. Platt Jr. et al. (2004) define a set of controllers suitable for this setting. They formulate this problem as an MDP and report preliminary learning experiments in which a ball is either moved to a certain location or is rotated 180 degrees. The nature of the manipulation task is that certain classes of controllers have to be executed in sequence to achieve an objective. The exact controllers used depend on the nature of the object being manipulated and the objective. But there is a higher level structure to the task that can be exploited to produce more efficient generalization. Before examining this structure, let us look at some of the classes of controllers used by Platt Jr. et al. (2004).

1. Localize, ${}^{L}\Phi$, uses the stereo vision system to locate a salient object, or a target position, in the work space. If the localize controller is formed by binding to a some feature of the object such as color or size, the resulting controller locates an object with that attribute.

2. Contact, ${}^{C}\Phi$, places the specified effector(s) in contact with the last localized object. The possible effectors are the left hand, l or the right hand, r or both, lr. Since the states of the underlying MDP are given by convergence states of

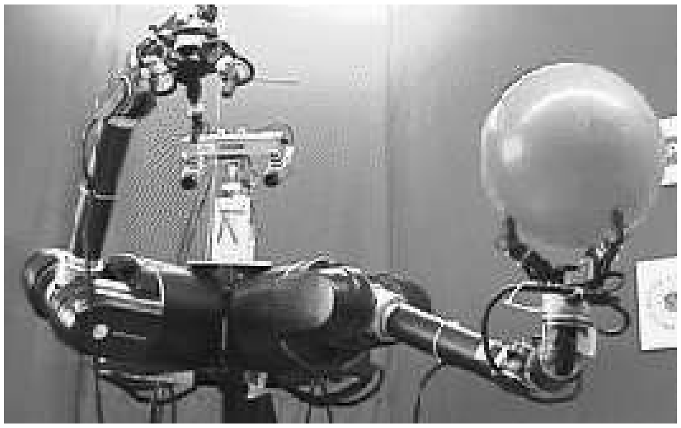


Figure 6.18. The UMass Torso holding a ball by opposing gravity with the left hand.

the controllers, the sensory resources available are also denoted by the same symbols as the controllers. Thus ${}^{C}\Phi_{l}^{l}$ denotes the controller that makes contact with an object using the left hand.

3. Grasp, ${}^{G}\Phi$, obtains a good grasp on the object the hand is currently in contact with. In the whole body grasping domain, controllers are formed by binding to a subset of one or both hands and gravity: $\{l, r, g\}$. For example, if the robot grasps an object by placing it on its left palm, it is modeled as holding it using the hand and gravity (Figure 6.18). The controller that achieves this configuration is then denoted by ${}^{G}\Phi_{l}^{lg}$. Since gravity is not under the control of the robot and it is not assigned as an effector.

4. Reach, ${}^{R}\Phi$, reaches to the last localized location. The sensors and effectors that can be used to form controllers are the same as with the grasp controller.

Additional controllers may be formed by suitably combining these basic controllers using the subject to constraint. Thus to move an object to a particular location while holding it with both hands, the robot would employ ${}^{R}\Phi_{lr}^{lr} \triangleleft {}^{G}\Phi_{lr}^{lr}$. Two grasp controllers can be combined, allowing the robot to change the grasp on an object. Thus to put

Localize	${}^{L}\Phi$			
Contact	${}^{C}\Phi_{l}^{l}$	${}^{C}\Phi_{r}^{r}$	${}^{C}\Phi_{lr}^{lr}$	
Grasp	${}^{G}\Phi_{lr}^{lr}$	${}^{G}\Phi_{l}^{lg}$	${}^{G}\Phi_{r}^{rg}$	${}^{G}\Phi_{lr}^{lrg}$
Reach	${}^{R}\Phi_{lr}^{lr}$	${}^{R}\Phi_{l}^{l}$	${}^{R}\Phi_{r}^{r}$	
Grasp ◁ Grasp	${}^{G}\Phi_{l}^{lg} \triangleleft {}^{G}\Phi_{lr}^{lr}$	${}^{G}\Phi_{r}^{rg} \triangleleft {}^{G}\Phi_{lr}^{lr}$	${}^{G}\Phi_{lr}^{lr} \triangleleft {}^{G}\Phi_{l}^{lg}$	${}^{G}\Phi_{lr}^{lr} \triangleleft {}^{G}\Phi_{r}^{rg}$
Reach ◁ Grasp	${}^{R}\Phi_{lr}^{lr} \triangleleft {}^{G}\Phi_{lr}^{lr}$	${}^{R}\Phi_{l}^{l} \triangleleft {}^{G}\Phi_{l}^{lg}$	${}^{R}\Phi_{r}^{r} \triangleleft {}^{G}\Phi_{r}^{rg}$	
	${}^{R}\Phi_{l}^{l} \triangleleft {}^{G}\Phi_{lr}^{lr}$	${}^{R}\Phi_{r}^{r} \triangleleft {}^{G}\Phi_{lr}^{lr}$		

Table 6.2. Subset of controllers for accomplishing two handed grasps of large objects.

an object that the robot is holding with both hands onto the left palm, it would employ ${}^{G}\Phi_{l}^{lg} \triangleleft {}^{G}\Phi_{lr}^{lr}$.

Employing this set of controllers the robot can manipulate a variety of objects in its workspace. As an example, the subset of controllers appropriate for two handed manipulation is shown in Table 6.2. The state of the system is described by a vector with one component for each controller. When actions are combined with subject to constraint, two or more bits of this representation are affected. A transition graph of a sub-task in the underlying MDP is shown in Figure 6.19. The task represented here is to localize on some object and, depending on its location and size, use the appropriate hand(s) to grasp it and move it to a specified target location. The left, central and right branches correspond to the cases when the object is grasped using the left hand, both hands and right hand respectively. Note that if the object is in the center and is small, it can be grasped using either the right or left hand. For large objects, both hands are needed. If a large object is in the right or left side of the workspace, the robot fails to grasp it.

Obviously there is a lot of redundancy in this representation. While different parameterization is needed for each branch in the graph, the over all structure of each path is the same: a contact, followed by a grasp and a reach. We can exploit

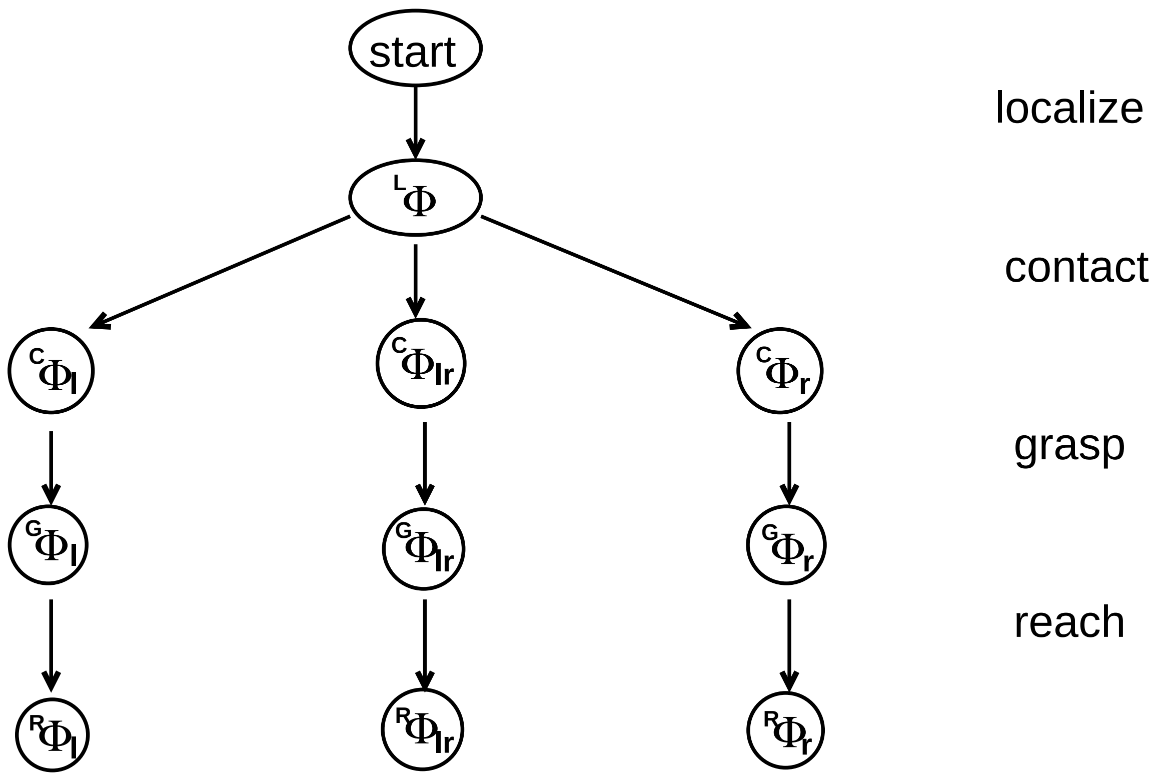


Figure 6.19. The transition structure of a family of sub-tasks for manipulating objects in the workspace. A state labeled ${}^{X}\Phi_{j}$ means that the controller ${}^{X}\Phi_{j}^{j}$ has converged in that state.

this structure to define a relativized option. The transition graph of the relativized option that models this family of sub-tasks, parameterized by the object size and location, is shown in Figure 6.20. The actions in the option MDP are the control objectives or potentials, without a specific parameterization. Similarly the state is represented by a vector with one component for each potential surface. The possible transformations from the original sub-task to this schema are the set of possible bindings to the basic potentials. The function $\psi(s)$ returns the size and location of the target object. Thus if there is a small object in the left side of the workspace, the correct transformation is to bind the sensor and effectors to l. For a large object in the center, the correct transformation is to bind to lr.

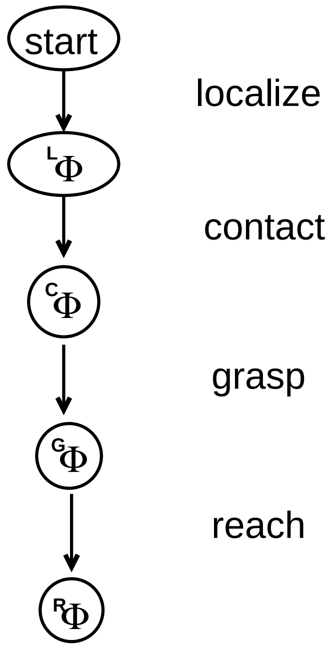


Figure 6.20. A relativized option that represents the family of tasks shown in Figure 6.19. A state labeled $^{X}\Phi$ means that for some j, the controller $^{X}\Phi_{j}^{j}$ has converged in that state.

The relativized option is derived under the implicit assumption that the entire workspace is reachable even holding an object with both hands. This is obviously not true in general. If the task is to pick up an object in the left side of the workspace and move it to right edge of the workspace, then the robot needs to transfer the left handed grasp on the ball to a right handed one before executing a reach. Similarly when picking up a large object and moving it to either edge of the field, a two handed grasp must be changed to a single handed grasp with gravity. As mentioned earlier this requires running a grasp controller subject to another grasp controller. This additional manipulation is not captured in the relativized option shown.

We would like to incorporate this additional maneuvering while still maintaining the transition structure shown in Figure 6.20, since it captures the underlying structure when the additional maneuvering is not required. Robert Platt suggests the following solution.[3] We introduce another relativized option with the transition structure shown in Figure 6.21. We distinguish between two kinds of grasps, good and bad. Good grasps are those that allow us to reach to the target location. The actions in this MDP are regrasp and reach. The reach action is the same as before. The regrasp action consists of controllers of the form ${}^{G}\Phi^{j}_{j'} \triangleleft {}^{G}\Phi^{k}_{k'}$, where j, j' and k, k' are different parameterizations for the grasp controller. There are three states: good grasp, poor grasp, and reach converged. The function $\psi(s)$ returns the target location, current location and size of the object. If no regrasping is required, the robot is in the good grasp state and transitions to the terminal state by executing a reach. If not the robot is in a poor grasp state and needs to execute the appropriate regrasp action to acquire a good grasp. This relativized option replaces the reach action in the option MDP in Figure 6.20.

We are currently working on validating this design on a simple task. The overall task is to clear a workspace of different sized balls, ranging from tennis balls to beach balls. There are two bins, one each at the left and right edges of the workspace. An operator indicates the target location, i.e., which of the bins into which a ball should be dropped. This task can be solved by repeatedly applying the relativized option shown in Figure 6.20. Note that since the transformations are different bindings to resources, effecting a change in the transformation is not as trivial as described in our earlier experiments. In order to change the transformation a sequence of regrasping actions have to be executed. Platt Jr. et al. (2004) address this issue in the context of learning to regrasp a given object.

6.7.3 Discussion

We have described the preliminary design of a very simple experimental demonstration of our ideas. This design is joint work with Robert Platt and Roderic Gru-

[3]In private discussion.

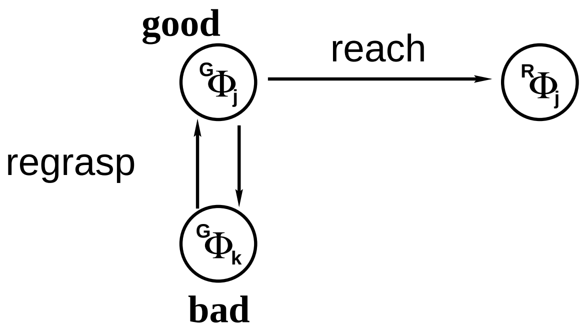


Figure 6.21. A relativized option that represents the *regrasp before reach* sub-task. A state labeled ${}^X\Phi_i$ means that the controller ${}^X\Phi_i^i$ has converged in that state, where i depends on the transformation chosen.

pen at the Laboratory for Perceptual Robotics at the University of Massachusetts, Amherst. Though conceptually it is straightforward, conducting even this experiment on the UMass Torso is not trivial. There are many issues to be addressed, including designing the software architecture, handling of error conditions, and implementing the transition choosing mechanism. With more experience on the real robot, the hierarchical decomposition could also be refined further. If successful, this work will be a first step in integrating the control basis architecture and our abstraction framework, specifically option schemas. This interface we feel would facilitate greater collaboration between the two approaches and lead to better synthesis of ideas. The abstraction notions developed in this work are generally applicable to complex problem domains. But in order to expand the applicability of our option schema framework we need to further enrich its capabilities. We discuss some promising directions in Chapter 7.

CHAPTER 7
CONCLUSIONS

7.1 Summary

In this thesis we introduced MDP homomorphisms as a paradigm for expressing various forms of abstractions in Markov decision processes. We showed that MDP homomorphisms are powerful and flexible enough to represent a wide variety of abstractions. We established theoretical guarantees as to the goodness of the abstract model formed. In particular we showed that an optimal solution of the reduced model can be used to induce an optimal solution for the original problem. This allows the definition of "ideal" abstraction, one that results in no loss of information relevant to solving the task at hand.

MDP homomorphism helps establish notions of equivalence—among states and actions and also between MDPs. The crucial innovation in our definition of MDP homomorphism is considering state-action equivalence. Much of the earlier work uses just state equivalence. Expanding the notion of equivalence allows one to model a wide class of abstractions as was established in this thesis.

Forming reduced models that preserve some aspect of the original system is the goal of model minimization algorithms. We develop a model minimization algorithm based on MDP homomorphisms. This algorithm is an extension of earlier work by Dean and Givan (1997). We also show that the notion of *stochastic bisimulation homogeneity*, which is the basis of the earlier framework, is equivalent to MDP homomorphism in the context of minimization. Employing state-action equivalence means that our minimization framework is strictly more powerful than Dean and Givan's.

We also extend a polynomial time minimization algorithm due to Lee and Yannakakis (1992) and Dean and Givan (1997) to our minimization framework.

In Chapter 4 we introduce the notion of symmetry groups of MDPs based on MDP homomorphisms and show that this can model symmetries of the system. This definition uses the mathematical structure of the MDP and does not rely on any special geometric properties of the system. While it can model usual notions of symmetry such as reflections, rotations etc, it is not limited to them. In particular, in problems with objects, symmetry groups can model symmetries arising for object interchangeability.

Factored MDPs are a compact paradigm for representing MDPs with structure. We explored certain forms of structured homomorphisms that can take advantage of the inherent redundancy and independence in a factored MDP representation. The minimal model of an MDP cannot usually be modeled by a structured homomorphic image. But searching for reduced models in a restricted space imposed by some structure is often the only feasible approach.

Factored symmetry groups do not necessarily lead to a smaller description of the reduced MDP, and a complete enumeration of the state space is required to derive the reduced model. We introduce a modification of an algorithm by Emerson and Sistla (1996) that uses MDP symmetry groups and constructs a reduced model incrementally without requiring complete enumeration of the state space.

The MDP homomorphism conditions are rather strict, and hence exact homomorphic equivalence is seldom obtained in practice. We introduce two notions of approximate homomorphisms derived from Whitt (1978) and Givan et al. (2000) that allow us to consider states that differ slightly in their dynamics as equivalent. Reduced models constructed under a relaxed notion of equivalence no longer guarantee the preservation of the optimality of solutions. We show how to bound the loss that result from approximations. These bound can be used as a guide in selecting appropriate approximations.

In Chapter 5 we develop a hierarchical decomposition framework that combines spatial abstraction with temporal abstraction. Partial SMDP homomorphisms form the basis for spatial abstraction in this framework. We extend the options framework (Sutton et al., 1999) to model temporal abstraction over Markov sub-goal tasks and spatial abstraction specific to the sub-task represented by the option. We introduce the notion of a relativized option that is a compact representation of a related family of family of tasks. We also show that the abstraction conditions developed earlier by Dietterich (2000a) for a related hierarchical framework are in fact a specialization of the more general SMDP homomorphism conditions. The utility of relativized options is empirically demonstrated on simple test beds.

In Chapter 6 we develop more sophisticated abstract representations based on relativized options. First we develop the concept of option schema, a prototype of an option, which is specified by an abstract state and action space. A learning agent can acquire skills or policies in this prototypical setting and then generalize them to other situations by suitably transforming this abstract space. We propose a Bayesian algorithm for selecting the right transformation to apply in a given setting and demonstrate that it is empirically correct. We apply option schemas to a complex game problem inspired by the the *Pengi* domain (Agre, 1988).

We also introduce deictic option schemas, where the class of permissible transformations applicable to the abstract space are defined via a set of pointers. Deixis is an indexical representation introduced to AI by Agre (1988). We show that under certain assumptions we can model methods that employ deictic representations as attempting to identify homomorphic reductions. We empirically demonstrate the utility of this view in a modification of the game domain.

MDP homomorphisms provide a formalism for expressing various abstractions and also guidelines for designing appropriate abstractions. Applying such abstractions in real world problems requires much design, guided by domain knowledge. We explore the design of an experimental setup involving a humanoid robot. We build on the control basis framework (Huber and Grupen, 1999; Huber, 2000) for hierarchical control and introduce suitable relativized options that enable efficient knowledge transfer at higher levels of the hierarchy.

7.2 Future Work

Efficient Minimization Algorithms

The minimization algorithms presented in this thesis do not explicitly exploit the structure of symmetry groups. As outlined in Chapter 4, partitions that arise from symmetry groups are structured. This is in addition to the structure modeled in factored MDPs. It is our contention that taking advantage of this structure would allow us to develop more efficient minimization algorithms.

The minimization algorithms presented in Chapters 3 and 4 assume the availability of a complete system model. While this assumption is valid in planning problems, a complete model is not available in typical learning formulations. The approach we take to forming abstraction is to leverage domain knowledge to define a family of transformations in which to search for homomorphisms. One could also incrementally construct a reduced image based on experience gained by interacting with the environment. McCallum (1995) and Jonsson and Barto (2001) approach the problem of abstraction along similar lines but use a different notion of equivalence. One useful direction of further research is to explore incremental construction techniques that employ MDP homomorphisms.

Automatic Discovery of Relativized Options

One of the chief difficulties in using relativized options, or any hierarchical RL approach, is designing an appropriate hierarchal decomposition. Autonomously discovering hierarchical structure is an active topic of research recently (Digney, 1998; McGovern and Barto, 2001; Hengst, 2002; Pickett and Barto, 2002). Many of these algorithms for hierarchy discovery require extensive experience, simulated or real. This experience can be used to construct a model of the environment and, in combination with a minimization algorithm, can be used to derive a reduced representation of the option discovered. This is a promising direction to pursue for constructing useful relativized options. The model itself would be an approximation of the true system dynamics and hence would introduce additional error while forming reduced models. We can use the approach due to Kearns and Singh (1998), specifically their *simulation lemma*, to characterize the accuracy of the constructed model.

Applications

This thesis lays the theoretical foundation for a flexible abstraction paradigm and provides tools for constructing powerful representational idioms. As we saw in Section 6.7, applying these ideas to specific domains requires significant amounts of domain knowledge and design. Relativized options and MDP homomorphisms provide us with tools for efficiently using this domain knowledge. Applications using the control basis framework are particularly suitable since much of the prior work that needs to be done in codifying the domain knowledge is in place and we can use our framework to build upon it. We desire to build general guidelines for exploiting domain knowledge in other application domains as well.

Deictic Representations

Memoryless consistent representation (CR) algorithms are a weak approach to learning with deictic representations. The only other memoryless CR algorithm, the

Lion algorithm, works only in deterministic environments. We can employ deictic option schemas in stochastic environments, but the assumption that the set of deictic pointers available is sufficient to express a CR is a restrictive assumption. In the Bayesian approach we employ in Chapter 5 we assume the existence of a fixed set of correct pointer configurations that do not change during the execution of a sub-task. This is seldom the case, especially when we assume that the transformations are implemented by physical sensors and effectors. In order to make deictic option schemas more flexible we need to allow the correct pointer configuration to change during a sub-task. One approach is to incorporate more features in the parameter estimation algorithm, which can evolve during the execution of a task. In addition to the current pointer values, using memory to derive the abstract representation also vastly enhances the power of deictic option schemas. More exploration is needed in this direction in order to verify the plausibility of these ideas.

Extending the Bayesian Algorithm

We formulate the problem of identifying the right transformation to apply in a given sub-task as a parameter estimation problem in which the agent chooses the right value from a discrete set of transformations. We can extend this approach to cases where the transformations are not known in advance, but a family of transformations is determined by some set of parameters. Such situations arise especially in domains with continuous system dynamics. The transformations may be determined by some continuous valued parameters, which can then be estimated using a Bayesian approach similar to the one presented in this thesis.

The algorithm presented in Chapter 5 considers only the transition dynamics. This is sufficient in many cases to identify the correct transformations, as demonstrated in the experiments. In some domains it becomes imperative to pay attention to the reward structure also in order to choose the correct transformations. One way to achieve this is to keep track of the *value* of each transformation along with the posterior probabilities. The values are updated using a stochastic approximation rule, but with the immediate rewards weighted by the current posteriors for the transformation. This is a simple intuitively appealing approach to incorporating reward structure in our search for the right transformation, and we are presently working on empirically validating it.

7.3 Closing Remarks

In this dissertation we developed an algebraic framework for describing abstraction in MDPs. One of the key insights in this work is that a wide variety of commonly used abstractions are in fact different aspects of the same underlying mathematical structure captured by MDP homomorphisms and symmetry groups of MDPs. This work is largely theoretical in nature, with limited empirical validation. In the later part of the dissertation we built powerful representational idioms, namely relativized options, option schemas and deictic option schemas, based on the notion of homomorphic equivalence. We envision these representational idioms forming part of the basis for lifelong learning in a situated agent. The agent would build a repertoire of schemas and transformations based on past training and experience. When confronted with a new situation the agent tries to use an existing schema under an appropriate transformation or acquire a new schema, building on its already acquired knowledge.

APPENDIX A
PROOF OF THEOREM 5

Definition: Let $h = \langle f, \{g_s | s \in S\}\rangle : \mathcal{M}_1 \to \mathcal{M}_2$ and $h' = \langle f', \{g'_s | s \in S\}\rangle : \mathcal{M}_2 \to \mathcal{M}_3$ be two MDP homomorphisms. The *composition* of h and h' denoted by $h \circ h'$ is a map from $\mathcal{M}_1$ to $\mathcal{M}_3$, with $(h \circ h')(s, a) = h'(h(s, a)) = \left(f'(f\,(s)), g'_{f(s)}(g_s\,(a))\right)$ for all $(s, a) \in \Psi$. It can be easily verified that $h \circ h'$ is a homomorphism from $\mathcal{M}_1$ to $\mathcal{M}_3$.

Theorem 5: Let B be the coarsest reward respecting SSP partition of MDP $\mathcal{M} = \langle S, A, \Psi, P, R\rangle$. The quotient MDP $\mathcal{M}/B$ is a minimal image of $\mathcal{M}$.

Proof: We will prove this by proving the contrapositive: if $\mathcal{M}/B$ is not a minimal image of $\mathcal{M}$, then B cannot be the coarsest reward respecting SSP partition of $\mathcal{M}$.

Let h be the homomorphism from $\mathcal{M}$ to $\mathcal{M}/B$. If $\mathcal{M}/B$ is not a minimal MDP, then there exists a homomorphism h' (that is not an isomorphism) from $\mathcal{M}/B$ to some MDP $\mathcal{M}'$. Therefore there exists a homomorphism $(h \circ h')$ from $\mathcal{M}$ to $\mathcal{M}'$. From the definition of composition, it is evident that $B_h < B_{(h \circ h')}$.

We need to show that B is not coarser than $B_{(h \circ h')}$. In other words we need to show that either $B < B_{(h \circ h')}$ or they are not comparable. From the construction of a quotient MDP it is clear that $B_h|S = B|S$ since we use $B|S$ as the states of $\mathcal{M}/B$. Since $\mathcal{M}'$ is a homomorphic image of $\mathcal{M}/B$ but is not isomorphic to it, either (i) $\mathcal{M}'$ has fewer states than $\mathcal{M}/B$ or (ii) some states in $\mathcal{M}'$ have fewer actions than $\mathcal{M}/B$. In case (i) we have that $B|S < B_{(h \circ h')}|S$. We know that this implies that B is not coarser than $B_{(h \circ h')}$. In case (ii) we have that $B|S = B_{(h \circ h')}|S$. Let $[s]_B \; (= [s]_{B_{(h \circ h')}})$

be a state with fewer admissible actions in $\mathcal{M}'$. This implies that s appears in fewer unique blocks in $B_{(h \circ h')}$ than in B. Thus $B < B_{(h \circ h')}$. Therefore B is not the coarsest reward respecting SSP partition. Hence $\mathcal{M}/B$ is a minimal image if B is the coarsest reward respecting partition of $\mathcal{M}$. $\square$

APPENDIX B

SOME RESULTS ON APPROXIMATIONS OF DYNAMIC PROGRAMS

In this appendix we present, without proofs, some results due to Whitt (1978) on error bounds in approximations of dynamic programs. Whitt develops these bounds for a general formulation of dynamic programs developed by Denardo (1967), known as the *monotone contraction operator model*. MDPs are a special case of such operator models. Much of the material presented in this section is derived from Whitt's (1978) paper.

B.1 Contraction Operator Model of Dynamic Programs

Definition: A *dynamic program* is defined by the tuple $\langle S, \{A_s, s \in S\}, h, \gamma \rangle$. Here S is the (non-empty) set of states and A_s is the set of actions admissible in state s. Let the policy space Π be the Cartesian product of the action spaces.[1] Let $\mathcal{V}$ be the set of all bounded real-valued functions on S, with the supremum norm: $\|V\| = \sup\{|V(s)| : s \in S\}$. The *local income function* h assigns a real number to each triple (s, a, V), with $s \in S$, $a \in A_s$ and $V \in \mathcal{V}$. The function h generates a return operator H_π on $\mathcal{V}$ for each $\pi \in \Pi$, with $[H_\pi(V)](s) = h(s, \pi(s), V)$. The following assumptions are made about the return operators:

(B) *Boundedness:* There exist numbers K_1 and K_2 such that $\|H_\pi V\| \leq K_1 + K_2\|V\|$ for all $V \in \mathcal{V}$ and $\pi \in \Pi$.

[1] This implies we are only considering deterministic policies.

(M) *Monotonicity:* If $V \geq U$ in $\mathcal{V}$, i.e., if $V(s) \geq U(s)$ for all $s \in S$, then $H_\pi V \geq H_\pi U$ in $\mathcal{V}$ for all $\pi \in \Pi$.

(C) *Contraction:* For $0 \leq \gamma < 1$, $\|H_\pi U - H_\pi V\| \leq \gamma \|U - V\|$, for all $U, V \in \mathcal{V}$ and $\pi \in \Pi$.

The contraction assumption implies that H_π has a unique fixed point in $\mathcal{V}$ for each $\pi \in \Pi$, denoted by V^π and called the return function of π. Let $V^\star$ denote the optimal return function defined by $V^\star(s) = \sup\{V^\pi(s) : \pi \in \Pi\}$. Let $H^\star$ be the maximization operator on $\mathcal{V}$ defined by $[H^\star(V)](s) = \sup\{[H_\pi(V)](s) : \pi \in \Pi\}$. The key property of this model is that the operator $H^\star$ also has the properties (B), (M) and (C) and it fixed point is the optimal value function $V^\star$.

To make this more accessible, let us see how this model can represent an MDP, $\mathcal{M} = \langle S, A, \Psi, P, R \rangle$. The states and action sets of the MDP are the states and actions of the corresponding dynamic program. The parameter γ has to be chosen apriori as is usually done. We define the local return function corresponding to $\mathcal{M}$ as follows:

$$h(s, a, V) = R(s, a) + \gamma \sum_{s'} P(s, a, s')V(s')$$

where V is a real valued function on S. If V corresponds to the value function for some fixed policy π, then the function h is in fact the Q function corresponding to the policy π. In traditional MDP notation, we restrict attention to only value functions corresponding to some policies and do not consider all elements of $\mathcal{V}$. Hence the traditional notation depends only on the policy and not on the function V.

For a fixed policy π, the operator H_π converges to the value function for the policy in $\mathcal{M}$. The operator models a technique known as iterative policy evaluation. The maximization operator $H^\star$ converges to the optimal value function for $\mathcal{M}$ and models the MDP solution technique known as *value iteration* (Puterman, 1994).

B.2 Whitt's Approximation Results

We present a brief description of Whitt's approximation framework for dynamic programs and state just the results we use in this thesis. For a more complete description and the proofs see (Whitt, 1978). Let $\mathcal{D} = \langle S, \{A_s, s \in S\}, h, \gamma \rangle$ and $\mathcal{D}' = \langle S', \{A'_{s'}, s' \in S'\}, h', \gamma' \rangle$ be two dynamic programs. We say $\mathcal{D}$ and $\mathcal{D}'$ are comparable and $\mathcal{D}'$ is an image of $\mathcal{D}$ is the following maps are well-defined: (1) a map f from S onto S', (2) a map g_s from A_s onto $A_{f(s)}$, (3) a map f' from S' into S such that $f(f'(s')) = s'$ for all $s' \in S'$ and (4) a map g'_s from $A_{f(s)}$ into A_s such that $g_s(g'_s(a')) = a'$ for all $a' \in A_{f(s)}$ and $s \in S$.

Given two comparable dynamic programs, the distance between them can be expressed in terms of the following quantity, for all $V \in \mathcal{V}$:

$$K(V) = \sup_{\substack{a \in A_s \\ s \in S}} |h(s, a, V) - h'(f(s), g_s(a), f(V))|$$

where $f(V) : S' \to \mathrm{I\!R}$ with $f(V)(s') = V(f'(s'))$ for each $s' \in S'$. We then define $K(V') = K(f'(V'))$ where $f'(V') : S \to \mathrm{I\!R}$ with $f'(V')(s) = V'(f(s))$ for each $s \in S$.

We state the following without proof:

Lemma 3.1: For all $U', V' \in \mathcal{V}'$, $|K(U') - K(V')| \leq (\gamma + \gamma') \|U' - V'\|$.

Corollary: If $\pi'^{\star}$ is an ϵ-optimal policy in Π' then:[2]

$$\left\| V^{\star} - V^{\pi'^{\star}_{\mathcal{D}}} \right\| \leq \frac{2}{1-\gamma} K(V^{\pi'^{\star}}) + \left(1 + \frac{\gamma + \gamma'}{1-\gamma}\right) \epsilon$$

where $\pi'^{\star}_{\mathcal{D}}$ is the policy formed by *lifting* $\pi'^{\star}$ to the dynamic program $\mathcal{D}$.[3]

[2]If π is an ϵ-optimal policy, then $\|V^{\star} - V^{\pi}\| \leq \epsilon$.

[3]Under Whitt's definition of lifting, we need to ensure that the lifted policy is also deterministic. Hence instead of assigning equal probabilities to all the pre-image actions, we just pick one.

The following result applies to stochastic sequential decision models like MDPs. For the sake of simplicity we present the Theorem as it applies to finite MDPs. In the previous section we saw how to construct a dynamic program to represent a MDP. Let us define the following quantities:

$$\begin{aligned}
K_r &= \max_{\substack{s\in S\\ a\in A_s}} \mid R(s,a) - R'(f(s), g_s(a)) \mid \\
K_p &= \max_{\substack{s\in S\\ a\in A_s}} \sum_{[s_1]_f \in B_f} \Big| T(s,a,[s_1]_f) - P'(f(s), g_s(a), f(s_1)) \Big| \\
\delta(V) &= \max_{s\in S} V(s) - \min_{s\in S} V(s) \\
\delta(V') &= \delta(f'(V')) \\
\delta_{r'} &= \max_{\substack{s'\in S'\\ a'\in A_{s'}}} R'(s',a') - \inf_{\substack{s'\in S'\\ a'\in A_{s'}}} R'(s',a')
\end{aligned}$$

Theorem 6.1 For any $V' \in \mathcal{V}'$, $K(V') \le K_r + \gamma\, \delta(V') \frac{K_p}{2}$.

Corollary: $K(V^{\pi'^\star}) \le K_r + \frac{\gamma}{1-\gamma} \delta_{r'} \, \frac{K_p}{2}$.

BIBLIOGRAPHY

Agre, P. E. (1988). The dynamic structure of everyday life. Technical Report AITR-1085, Massachusetts Institute of Technology.

Agre, P. E. and Chapman, D. (1987). Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*.

Amarel, S. (1968). On representations of problems of reasoning about actions. *Machine Intelligence*, 3:131–137.

Andre, D. and Russel, S. (2001a). Programmable reinforcement learning agents. In Dietterich, T. G., Tresp, V., and Leen, T. K., editors, *Proceedings of Advances in Neural Information Processing Systems 13*, pages 1019–1025, Cambridge, MA. MIT Press.

Andre, D. and Russel, S. J. (2001b). State abstraction for programmable reinforcement learning agents. Submitted to NIPS 2001.

Arbib, M. A. (1995). Schema theory. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 830–834. MIT Press, Cambridge, MA.

Arbib, M. A. and Manes, E. G. (1975). *Arrows, Structures and Functors.* Academic Press, New York, NY.

Ballard, D. H., Hayhoe, M. M., Pook, P. K., and Rao, R. P. N. (1996). Deictic codes for the embodiment of cognition. Technical Report NRL95.1, University of Rochester.

Bartlett, F. C. (1932). *Remembering.* Cambridge University Press, Cambridge England.

Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA.

Boutilier, C., Dean, T. L., and Hanks, S. (1999). Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94.

Boutilier, C. and Dearden, R. (1994). Using abstractions for decision theoretic planning with time constraints. In *Proceedings of the AAAI-94*, pages 1016–1022. AAAI.

Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. In *Proceedings of International Joint Conference on Artificial Intelligence 14*, pages 1104–1111.

Boutilier, C., Reiter, R., and Price, R. (2001). Symbolic dynamic programming for first-order MDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 541–547.

Bradtke, S. J. and Duff, M. O. (1995). Reinforcement learning methods for continuous-time Markov decision problems. In Minton, S., editor, *Advances in Neural Information Processing Systems*, volume 7, pages 393–400, Cambridge, MA. MIT Press.

Chapman, D. (1991). *Vision, Instruction, and Action.* MIT Press, Cambridge, MA.

Cleary, M. E. (1997). *Systematic use of Deictic Commands for Mobile Robot Navigation.* PhD thesis, Northeastern University, Boston, MA, USA.

Coelho, J. A. and Grupen, R. A. (1997). A control basis for learning multifingered grasps. *Journal of Robotic Systems*, 14(7):545–557.

Dayan, P. and Hinton, G. E. (1993). Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5*, pages 271–278. Morgan Kaufmann.

Dean, T. and Givan, R. (1997). Model minimization in Markov decision processes. In *Proceedings of AAAI-97*, pages 106–111. AAAI.

Dean, T., Givan, R., and Kim, K.-E. (1998). Solving planning problems with large state and action spaces. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems.*

Dean, T., Givan, R., and Leach, S. (1997). Model reduction techniques for computing approximately optimal solutions for Markov decision processes. In *Proceeding of UAI-97.*

Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computer Intelligence*, 5(3):142–150.

Dean, T. and Lin, S.-H. (1995). Decomposition techniques for planning in stochastic domains. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence.*

Denardo, E. V. (1967). Contraction mappings in the theory underlying dynamic programming. *SIAM Review*, 9:165–177.

Dietterich, T. G. (1998). Hierarchical reinforcement learning with the MAXQ value function decomposition. In *Proceedings of the 15th International Conference on Machine Learning ICML'98*, San Mateo, CA. Morgan Kaufmann.

Dietterich, T. G. (2000a). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Artificial Intelligence Research*, 13:227–303.

Dietterich, T. G. (2000b). An overview of MAXQ hierarchical reinforcement learning. In Choueiry, B. Y. and Walsh, T., editors, *Proceedings of the Fourth Symposium on Abstraction, Reformulation and Approximation SARA 2000, Lecture Notes in Artificial Intelligence*, pages 26–44, New York, NY. Springer-Verlag.

Dietterich, T. G. (2000c). State abstraction in MAXQ hierarchical reinforcement learning. In Solla, S. A., Leen, T. K., and Muller, K., editors, *Proceedings of Advances in Neural Information Processing Systems 12*, pages 994–1000, Cambridge, MA. MIT Press.

Digney, B. (1996). Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In Maes, P. and Mataric, M., editors, *From animals to animats 4: The fourth conference on the Simulation of Adaptive Behavior SAB 96*. MIT Press/Bradford Books.

Digney, B. (1998). Learning hierarchical control structure for multiple tasks and changing environments. In *From animals to animats 5: The fifth conference on the Simulation of Adaptive Behavior: SAB 98*.

Drescher, G. L. (1991). *Made-Up Minds*. MIT Press, Cambridge, MA.

Drummond, C. (1998). Composing functions to speed up reinforcement learning in a changing world. In *European Conference on Machine Learning*, pages 370–381.

Emerson, E. A., Jha, S., and Peled, D. (1997). Combining partial order and symmetry reductions. In Brinksma, E., editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 19–34, Enschede, The Netherlands. Springer Verlag, LNCS 1217.

Emerson, E. A. and Sistla, A. P. (1996). Symmetry and model checking. *Formal Methods in System Design*, 9(1/2):105–131.

Emerson, E. A. and Sistla, A. P. (1997). Utilizing symmetry when model-checking under fairness assumptions: An automata-theoretic approach. *ACM Transactions on Programming Languages and Systems*, 19(4):617–638.

Emerson, E. A. and Trefler, R. J. (1998). Model checking real-time properties of symmetric systems. In *Mathematical Foundations of Computer Science*, pages 427–436.

Emerson, E. A. and Trefler, R. J. (1999). From asymmetry to full symmetry: New techniques for symmetry reduction in model checking. In *Conference on Correct Hardware Design and Verification Methods*, pages 142–156.

Evans, S. H. (1967). A brief statement of schema theory. *Psychonomic Science*, 8:87–88.

Feng, Z., Hansen, E. A., and Zilberstein, S. (2003). Symbolic generalization for on-line planning. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI 2003)*.

Finney, S., Gardiol, N. H., Kaelbling, L. K., and Oates, T. (2002). That thing we tried didn't work very well: Deictic representation in reinforcement learning. In *Proceedings of the 18th International Conference on Uncertainty in Artificial Intelligence*.

Getoor, L., Friedman, N., Koller, D., and Pfeffer, A. (2001). Learning probabilistic relational models. In Dzeroski, S. and Lavrac, N., editors, *Relational Data Mining*. Springer-Verlag.

Givan, R. and Dean, T. (1997). Model minimization, regression, and propositional strips planning. In *Proceedings of the International Joint Conference on Artificial Intelligence '97*.

Givan, R., Dean, T., and Greig, M. (2003). Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223.

Givan, R., Leach, S., and Dean, T. (2000). Bounded-parameter Markov decision processes. *Artificial Intelligence*, 122:71–109.

Glover, J. (1991). Symmetry groups and translation invariant representations of Markov processes. *The Annals of Probability*, 19(2):562–586.

Hartmanis, J. and Stearns, R. E. (1966). *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, Englewood Cliffs, NJ.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. In *Proceedings of the 19th International Conference on Machine Learning*, pages 243–250.

Hennessy, M. and Milner, R. (1985). Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 31(1):137–161.

Hernandez-Gardiol, N. and Mahadevan, S. (2001). Hierarchical memory-based reinforcement learning. In Dietterich, T. G., Tresp, V., and Leen, T. K., editors, *Proceedings of Advances in Neural Information Processing Systems 13*, pages 1047–1053, Cambridge, MA. MIT Press.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes.* MIT press, Cambridge, MA.

Howarth, R. J. and Buxton, H. (1993). Selective attention in dynamic vision. In *Proceedings of the Thirteenth IJCAI Conference*, pages 1579–1584.

Huber, M. (2000). *A Hybrid Architecture for Adaptive Robot Control.* PhD thesis, University of Massachusetts, Amherst, Massachusetts, USA.

Huber, M. and Grupen, R. A. (1999). A hybrid architecture for learning robot control tasks. In *Proceedings of the 1999 AAAI Spring Symposium Series: Hybrid Systems and AI: Modeling, Analysis and Control of Discrete + Continuous Systems*, Stanford University, CA.

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3:285–317.

Ip, C. N. and Dill, D. L. (1996). Better verification through symmetry. *Formal Methods in System Design*, 9(1/2).

Jaakkola, T., Jordan, M. I., and Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201.

Jarvella, R. J. and Klein, W. (1982). *Speech, place, and action: studies in deixis and related topics.* John Wiley & Sons Ltd.

Jonsson, A. and Barto, A. G. (2001). Automated state abstraction for options using the u-tree algorithm. In Dietterich, T. G., Tresp, V., and Leen, T. K., editors, *Proceedings of Advances in Neural Information Processing Systems 13*, pages 1054–1060, Cambridge, MA. MIT Press.

Jump, J. R. (1969). A note on the iterative decomposition of finite automata. *Information and Control*, 15:424–435.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134.

Kearns, M. and Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. In *Proceedings of the 15th International Conference on Machine Learning*, pages 260–268. Morgan Kaufmann, San Francisco, CA.

Kemeny, J. G. and Snell, J. L. (1960). *Finite Markov Chains.* Van Nostrand, Princeton, NJ.

Kim, K.-E. and Dean, T. (2001). Solving factored MDPs via non-homogeneous partitioning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 747–752.

Knoblock, C. A. (1990). Learning abstraction hierarchies for problem solving. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, volume 2, pages 923–928, Boston, MA. MIT Press.

Koller, D. and Parr, R. (2000). Policy iteration for factored MDPs. In *Proceedings of the Sixteenth Uncertainty in AI Conference*, pages 326–334.

Koller, D. and Pfeffer, A. (1998). Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 580–587, Madison, WI. AAAI Press.

Land, M. F., Mennie, N., and Rusted, J. (1998). Eye movements and the roles of vision in activities of daily living: making a cup of tea. *Investigative Ophthalmology and Visual Science*, 39(S457).

Lang, S. (1967). *Algebraic Structures*. Addison Wesley, Reading, MA.

Larsen, K. G. and Skou, A. (1991). Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28.

Lee, D. and Yannakakis, M. (1992). Online minimization of transition systems. In *Proceedings of* 24th *Annual ACM Symposium on the Theory of Computing*, pages 264–274. ACM.

Lyons, D. M. and Arbib, M. A. (1989). A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5:280–293.

McCallum, A. (1995). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, Computer Science Department, University of Rochester.

McGovern, A. and Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning ICML 2001*, pages 361–368.

McMillan, K. L. (1993). *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell Massachusetts.

Minut, S. and Mahadevan, S. (2001). A reinforcement learning model of selective visual attention. In *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal.

Parr, R. (1998). *Hierarchical Control and learning for Markov decision processes*. PhD thesis, University of California at Berkeley.

Parr, R. and Givan, R. (2001). Large state space techniques for Markov decision processes. Invited presentation at Dagstuhl Seminar 01451, Exploration of Large State Spaces.

Parr, R. and Russell, S. (1997). Reinforcement learning with hierarchies of machines. In *Proceedings of Advances in Neural Information Processing Systems 10*, pages 1043–1049. MIT Press.

Paz, A. (1971). *Introduction to Probabilistic Automata*. Academic Press, New York, NY.

Piaget, J. (1952). *The Origins of Intelligence in Children*. International Universities Press.

Piaget, J. (1954). *The Construction of Reality in the Child*. Basic Books.

Pickett, M. and Barto, A. G. (2002). Policy blocks: An algorithm for creating useful macro-actions in reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*.

Platt Jr., R., Fagg, A. H., and Grupen, R. A. (2003). Extending fingertip grasping to whole body grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation*.

Platt Jr., R., Fagg, A. H., and Grupen, R. A. (2004). Manipulation gaits: Sequences of grasp control tasks. Submitted to the IEEE International Conference on Robotics and Automation.

Popplestone, R. and Grupen, R. (2000). Symmetries in world geometry and adaptive system behaviour. In *Proceedings of the 2nd International Workshop on Algebraic Frames for the Perception-Action Cycle (AFPAC 2000)*, Kiel, Germany.

Precup, D. (2000). *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst.

Puterman, M. L. (1994). *Markov Decision Processes*. Wiley, New York, NY.

Russel, S. and Norvig, P. (1995). *Artificial Intelligence - A Modern Approach*. Prentice-Hall, Englewood Cliffs.

Satia, J. K. and Lave, R. E. (1973). Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21:728–740.

Schmidt, R. A. (1975). A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225–260.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning. An Introduction*. MIT Press, Cambridge, MA.

Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211.

Sweeny, J. D. (2003). Active QoS flow maintenance in robotic, mobile, ad hoc networks. Synthesis Project Report, University of Massachusetts, Amherst.

Ullman, S. (1984). Visual routines. *Cognition*, 18:97–159.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* PhD thesis, Cambridge University, Cambridge, England.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical note: q-learning. *Machine Learning*, 8(3/4):279–292.

White, C. C. and Eldeib, H. K. (1986). Parameter imprecision in finite state, finite action dynamic programs. *Operation Research*, 34:120–129.

White, C. C. and Eldeib, H. K. (1994). Markov decision processes with imprecise transition probabilities. *Operations Research*, 43:739–749.

Whitehead, S. D. and Ballard, D. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7:45–83.

Whitehead, S. D. and Lin, L.-J. (1995). Reinforcement learning of non-Markov decision processes. *Artificial Intelligence*, 73:271–306.

Whitt, W. (1978). Approximations of dynamic programs I. *Mathematics of Operations Research*, 3(3):231–243.

Zeigler, B. P. (1972). On the formulation of problems in simulation and modelling in the framework of mathematical system theory. In *Proceedings of the Sixth International Congress on Cybernetics*, pages 363–385. Association Internationale de Sybernétique.

Zinkevich, M. and Balch, T. (2001). Symmetry in Markov decision processes and its implications for single agent and multi agent learning. In *Proceedings of the 18th International Conference on Machine Learning*, pages 632–640, San Francisco, CA. Morgan Kaufmann.