Predatory Sequence Learning for Synthetic Characters

by

Matthew Roberts Berlin
A.B. Computer Science, Harvard College, 2001

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2003

Author ....
Program in Media Arts and Sciences
August 8, 2003

Certified by ....
Bruce M. Blumberg
Associate Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by ....
Andrew B. Lippman
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Predatory Sequence Learning for Synthetic Characters

by

Matthew Roberts Berlin

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on August 8, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

The process of mammalian predatory sequence development offers a number of insights relevant to the goal of designing synthetic characters that can quickly and easily learn complicated and interesting behavior. We propose a number of principles for designing such learning systems, inspired by a targeted review of animal developmental phenomena, with particular emphasis on the development of predatory behavior in certain felid and canid species. We describe the implementation of a few of these principles as an extension to a popular algorithm for learning in autonomous systems called hierarchical Q-learning. In this new approach, the agent starts out with only one skill, and then new skills are added one at a time to its available repertoire as time passes. The agent is motivated to experiment thoroughly with each new skill as it is introduced. Simulation results are presented which empirically demonstrate the advantages of this new algorithm for the speed and effectiveness of the learning process.

Thesis Supervisor: Bruce M. Blumberg
Title: Associate Professor of Media Arts and Sciences

Predatory Sequence Learning for Synthetic Characters

by

Matthew Roberts Berlin

The following people served as readers for this thesis:

Thesis Reader ....
Deb Roy
Assistant Professor of Media Arts and Sciences
Program in Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader ....
Raymond Coppinger
Professor of Biology
Hampshire College

Acknowledgments

This thesis would not have been possible without the support of a great many people. Many thanks to my advisor Bruce Blumberg, who has given me a wonderful home here at the Media Lab for the past four years, and whose guidance has been pivotal in shaping my research and this thesis in particular. Thanks also to my readers Deb Roy and Raymond Coppinger for providing insightful feedback and for putting up with my everything-at-the-last-minute tendencies.

I am particularly indebted to Jesse Gray, who has been making things better for me for many years now. This thesis probably wouldn't have happened without his help, and it certainly wouldn't have been very good.

Bill Tomlinson and Marc Downie got me started down this path. I have been tremendously influenced by their energy and intelligence, and I am very grateful for their friendship.

I have had the great fortune to be surrounded by ridiculously cool and blisteringly intelligent colleagues. It has been a true pleasure working with Derek Lyons, Daphna Buchsbaum, Matt Grimes, Lily Shirvanee, Aileen Kawabe, Jennie Cochran, Seth Block, and Lauren McHugh in the Synthetic Characters group. I have also had the honor of working with a number of other great minds over the years, in particular Mike Johnson, Yuri Ivanov, Damian Isla, Rob Burke, Ben Resner, and Scott Eaton. Further thanks are due to Ari Benbasat, Josh Lifton, and Dan Stiehl, whose friendship has benefitted me greatly.

Many, many, many thanks to Rachel Kline. Your love, support, and patience throughout this whole process has meant the world to me.

This thesis is dedicated to my family: my parents Rachel and Richard, and my brother Alexander. Mom, Dad, Alex - you gave me strength when I needed it most over these past few months, and you continue to do so. I love you all so much, and carry you in my heart always.

Contents

Chapter 1

Introduction

The process of predatory behavior development in certain felid (cat family) and canid (dog family) species offers a number of insights relevant to the goal of designing computational systems that can quickly and easily learn complicated and interesting behavior. In the Synthetic Characters group, our goal is to create real-time, interactive, graphical creatures that behave and learn like real animals. The most sophisticated creature that we have created to date is Dobie T. Coyote, who can be trained to respond to vocal commands and who can learn new motor skills through a process known as luring [4, 3]. While these are both interesting and complicated forms of learning, Dobie does not yet have the ability to learn coordinated sequences of actions for satisfying his goals. We believe that this ability represents an important step towards making the behavior of synthetic characters seem more intelligent and realistic. My research confronts this problem of sequence learning by looking closely at one example of how the problem gets solved in nature: predatory sequence development.

The predatory sequence is an important part of the behavior of many different felid and canid species. In a typical hunting session, a wild cat will exhibit a number of different actions in sequence. First it will approach, then stalk, pounce, grab, kill, and ultimately eat its prey. Performing each action in the sequence at the right time and in the right context is critical for the survival of the animal. While the stakes are very high and the learning problem involved is very difficult, nevertheless these animals learn to hunt seemingly effortlessly during the first few months of life. It is this effortless learning of complicated behavior that we would like to capture, and add to the functionality of our synthetic creatures.

My recent research has proceeded along two major lines of inquiry, centered around two essential questions. First, what do predatory canids and felids actually learn during their individual development, and how do they learn it? Second, how are the elements of animal developmental learning interesting for computer science and for the design of new learning algorithms that improve upon existing techniques?

The first question has been the subject of a great deal of excellent research by biologists and experimental psychologists. My exploration of this literature has led me to a number of observations which have challenged some of my fundamental assumptions about animal learning. As a general point, animal development is a much more structured process than I had originally expected, with much more information encoded prior to learning. In particular, I had expected to find that the most important part of predatory sequence development was the learning of the actual ordering of the behavioral sequence. As it turns out, the ordering of individual behaviors seems to be largely specified ahead of time. Predatory sequence learning turns out to be much less about the actual structure of the sequence, and much more about how the predatory system as a whole gets motivated, released by external stimuli, and integrated with other important behavior systems such as escape and sexual behavior.

The second question, that of the relevance of animal development to computer science, defines a broad new area of research, one that my colleagues and I hope will keep us busy for quite some time. This thesis proposes some preliminary answers to this question, and introduces a new learning algorithm inspired by animal development. Ultimately, we hope that our research into such algorithms may someday give something back to the biology it is based upon, providing some new insights into the underlying developmental phenomena.

1.1 Road Map

The remainder of this thesis is organized as follows. Chapter 2 provides a focused review of animal development, with particular attention to the development of predatory behavior in certain felid and canid species. Chapter 2 also introduces a number of lessons inspired by the natural phenomena for designing learning systems, and discusses some of the implications of these lessons for technology and for the research of the Synthetic Characters group.

Chapter 3 discusses Q-learning and hierarchical Q-learning, popular algorithms for learning in autonomous agents, and examines how the lessons introduced in Chapter 2 can be incorporated into this paradigm, resulting in the creation of a new learning algorithm. Simulations involving this new algorithm are described in Chapter 4, and results are presented demonstrating the advantages and disadvantages of this technique in terms of the speed and effectiveness of the learning process.

Concluding thoughts and some ideas about future directions for this research are presented in Chapter 5.

Chapter 2

Development

Why is animal development interesting for computer science? This chapter attempts to answer this question. I discuss a number of potential applications for computer systems that develop like animals. I then propose three design principles inspired by observations from the behavioral literature which I hope will facilitate the creation of these new technologies. Finally, I introduce a number of ideas for directly applying these principles to the current practice of synthetic character design. These ideas ultimately form the basis for a new learning algorithm discussed in the following chapters.

2.1 Potential Applications for Developing Technology

What is the use of technology inspired by animal development? What does it mean in the long term to apply lessons from animal development to the design of synthetic creatures? I believe that the ideal application for developing technology is one where you want multiple integrated sequences of behavior, but where it is hard to specify either the motor domain, the perceptual environment, or the underlying control system ahead of time. Development can provide a powerful framework for filling in the required information through experience, especially in situations where the system is expected to satisfy multiple different goals simultaneously.

In addition to situations where you want to explicitly model the developmental process, such as in virtual pets or video-game environments, developmental learning might be directly useful for a number of other interesting applications. In particular, developmental technology enables us to build systems that can engage in context triggered learning, effectively recognizing when the environment is ready to provide them with the information that they need to learn what they need to learn.

The goal, ideally, is to build systems that learn when you're ready to teach them. Such systems could operate by identifying a caregiver or other trusted source of information in the environment, and then recognizing when that information source is ready to provide them with what they need to know. They might incorporate a set of learning specialists that turn on and off when relevant information becomes available in the environment, and might even engage in exploratory behaviors to help gauge the availability of such information.

As one example, such systems might be useful for dealing with a noisy robot manufacturing process. If a robot factory produces sensors with noisy specifications or failure-prone hardware, then the robots might be able to overcome these deficits through developmental learning. They could bootstrap their behavior while learning in a coordinated fashion about their own perceptual and physical abilities, bridging the gap between the expectations of their control software and the realities of their hardware. Such technology could thus serve to make robot manufacturing more cost-effective.

Developmental learning may be particularly important for designing a robot for an unknown of hostile environment, such as for deep-sea exploration or Mars exploration. Since the effects of the environment on the robot's motors might be difficult to fully predict, motor parameters might need to be calibrated on the fly, and the timing and coordination of motor skills might need to be learned as well. Search templates for releasing specific behaviors could be refined through experience and direct experimentation with the environment. Animal development provides a useful framework for thinking about how best to carry out and integrate these multiple learning tasks.

Ultimately, the goal is to create of a new breed of intelligent systems, virtual creatures whose learning is intimately connected with their natural behavior and interactions with their environment. We seek to build creatures who can learn while still satisfying their fundamental goals in a consistent fashion, and who know not only how to learn, but what to learn and when, and just as importantly when to stop learning. I contend that we can achieve these design goals not by reinventing the wheel, but by looking closely at systems that elegantly achieve these goals in nature, systems such as cats and dogs that can quickly and easily learn complicated and interesting behavior.

2.2 Lessons for Designing Learning Systems

In this section, I propose three lessons drawn from the study of animal development that are of significance for synthetic character design and for the design of intelligent systems more generally. Along the way, I review the observations from the behavioral literature which helped to provoke and shape these lessons.

While a fair number of sources are cited in this section, the core ideas were inspired to a large extent by the Coppingers' descriptions and analyses of dog behavior [6] and the observations of cat predatory behavior recorded by Leyhausen [17] and Baerends-van Roon and Baerends [1]. Most of the other papers cited appear in the excellent reader on the development of animal behavior put together by Bolhuis and Hogan [5].

Animal behavior is fundamentally local, varying widely between different species and different environmental niches. The discussion in this section focuses on a number of specific observations of particular species, and is not intended to generalize across broad swaths of the animal kingdom. In focusing on these specific observations, my aim is rather to reflect upon and hopefully inform the design of synthetic learning systems. As a related point, please note that while I use the term "animal" throughout this thesis for convenience, my conclusions are primarily based on observations of the specific species discussed in this section.

The study of animal development offers a number of fundamental lessons for designers of intelligent learning systems. From my initial research in this area, I propose the following three lessons.

First, the question when designing a learning system is not so much how to learn, but what to learn and when. A great deal of excellent work has been done in computer science on learning mechanisms, essentially addressing the question of how to learn, but much less work has been done on the questions of what to learn and when. These questions seem to be as important, if not more important, for intelligent systems in the real world.

The second lesson is to learn things in the right order. Instead of trying to learn everything all at once, animals seem to learn one thing at a time, confronting very specific different learning problems at different stages in their development.

The third and final lesson is that targeted motivation and reward signals that are matched to the specific learning task at hand can greatly improve the learning process. As cats develop, for example, the specific learning tasks that they are engaged in change, and at the same time the motivation and reward signals that they seem to be attending to also change. This seems to have the effect of effectively channeling information from the environment to the specific components of the behavior that the information pertains to.

Taken together, these lessons support the contention that prior knowledge is not a bad thing. Natural learning almost always leverages a rich set of expectations, and the ability to easily and safely incorporate such prior knowledge into synthetic learning systems should be a highly sought-after goal.

These lessons reflect a wide array of observations of real animal behavior, suggesting that certain properties of learning are shared by a number of different species. This is interesting, but does not in itself justify their relevance to computer science. The fact that a number of highly successful learning systems (animals) share a particular property does not necessarily mean that that property is causal to their success. The justification for all of these lessons ultimately lies in the advantages that they offer for machine learning. These advantages will be explored further throughout the rest of this thesis, both theoretically and in some cases experimentally.

It should also be noted that these lessons could have been arrived at through an analysis of synthetic learning systems directly, without ever having looked at animal behavior. However, I firmly believe that my focus on animal development greatly facilitated the formulation of these lessons. Furthermore, animal development provides a rich framework for understanding and contextualizing these lessons, as well as for reasoning about how to implement them synthetically. Hopefully, it will also make it easier to extrapolate from them to powerful new design principles in the future. Animals are remarkably capable learners, and by identifying more fully the characteristics of their success, we can arrive at a better understanding of our own goals as designers.

We proceed by examining each of these lessons in greater detail, highlighting observations from animal development that support these design principles and discussing their applicability to computational learning systems.

2.2.1 Not how to learn, but what to learn and when.

The question of what to learn and when is often just as important as the question of how to learn, but it has received much less attention in computational learning research. Timing is everything in animal learning. Animal learning throughout development is not a continuous process, but rather highly discrete, consisting of overlapping episodes of highly specialized learning. Animals learn very different things at different points in their development. They are adept at learning when the environment is ready to teach, as if they were "looking for" the requisite information when it is most likely to be available and easiest to acquire. As an example, many different species learn about their allies while in the nest, under the protective custody of their mothers, and subsequently learn about their enemies when they leave or begin to venture out of the nest. In a similar vein, rats learn to be choosy about their food when they start finding it for themselves, and not while they are suckling, when their only available food source is their mother's milk [10].

The developing animal brain seems to be a system of numerous learning specialists that turn on and off at remarkably appropriate moments during the creature's life. The underlying phenomenon behind this system of specialists is the fairly well-known idea of the sensitive period. Sensitive periods were originally termed "critical periods," and the original concept was of a critical period for certain types of learning, characterized by a sudden onset, a sudden offset, and by the irreversibility of learning. Each of these three characteristics has been the subject of significant debate, leading to the more recent, less committed idea of a "sensitive period": a period of augmented sensitivity to a particular type of experience, with a potentially gradual onset and offset, and with consequences that can sometimes be changed as a result of experience later in life. The onset and offset of a sensitive period do not have to be controlled by the same mechanism. For example, the onset of a particular sensitive period might be timed by an internal clock, whereas the offset might be based on experience [2]. While the details may be controversial, the core idea is not generally contested: animals are often particularly sensitive to certain types of experience during specific periods in their development.

The effects of sensitive periods can sometimes be overcome through experience later in life, but this is often difficult and often only occurs under special circumstances, such as an extreme deficiency of relevant stimuli during the sensitive period [2]. When learning does occur later in life that is similar to the learning that occurred or should have occurred during the sensitive period, an important question to ask is whether or not the learned representation is really the same. Similarity in the observable behavior does not necessarily imply similarity in the underlying mental representation. When a human adult learns a second language, they often end up speaking in much the same way as someone who learned that language as a child, but the learning process that is required is significantly different, and it is probable that the underlying representation is significantly different as well.

It is interesting to note that many species seem to have two or more different channels for developing important elements of their behavior. In one channel, the behavior develops under normal circumstances with a strong grounding in experience. In another channel, a highly stylized form of the behavior seems to develop as a fail-safe in the absence of the necessary experience.

An interesting example of the importance of sensitive periods for the development of sophisticated behavior comes from predatory behavior development in dogs. For dogs, much of the plasticity in their predatory behavior seems to involve which types of objects and creatures they will direct their predatory behaviors towards, and in what contexts these behaviors will be exhibited. Coppinger and Coppinger suggest that a dog's environmental and behavioral context during its critical period for socialization plays a key role in determining its adult behavior. For example, dogs for whom the onset of predatory behavior occurs after the critical period for socialization will not prey upon livestock with whom they have been raised. The Coppingers theorize that such details of developmental timing, in conjunction with small differences in the innate structure of the predatory sequence, may explain the significant differences between the observable behavior of different breeds of dogs, such as sheep-herding dogs, pointers, retrievers, and livestock-guarding dogs. [6]

2.2.2 Learn things in the right order.

Hand in hand with the lesson that the question of what to learn and when is critical for the development of interesting behavior comes another important lesson: learn things in the right order. Instead of trying to learn a sophisticated behavior all at once, animals break down this complicated learning task. They tend to learn behaviors one piece at a time, confronting very specific different learning problems at different stages in their development. For many different behaviors across many different species, these learning problems seem to follow a sort of precedence or natural ordering.

In the typical developmental timeline for a cat, the first thing that happens is that the individual motor patterns of the predatory sequence appear "prefunctionally," which is to say that they are expressed individually, outside of the context of the completed sequence, and appear apart from their ultimate functional purpose of providing the cat with food. This behavior starts to appear around postnatal day 22, and typically involves foreleg slamming and biting directed towards littermates instead of prey such as mice. [1]

New motor elements continue to appear through the end of the fourth week. After this point, development seems to consist mainly of adaptive modifications and combinations of existing motor patterns. As this occurs, the physical form of the behaviors improves, and the behaviors come to more strongly resemble their adult forms. During weeks five and six, fighting becomes integrated with locomotion: walking, running, jumping away and towards. [1]

After the improvements to the physical form of the behaviors have begun, their orientation and timing begins to improve with respect to external stimuli. Next, the predatory sequence as a whole begins to gel. Through the end of week eight, the transitions between the individual elements become visibly smoother, and the entire behavior proceeds more rapidly. [1]

Finally, after organization has improved within the predatory behavior system, organization begins to improve between different behavior systems, as the predatory sequence as a whole becomes more smoothly integrated with other important behavior systems and integrated into the creature's full behavioral repertoire. Most importantly, the predatory system becomes more strongly integrated with the inhibitory influence of the creature's escape system. Baerends-von Roon and Baerends argue that the integration of the attack and escape systems may result in the appearance of a number of important behaviors such as stalking, as the result of inhibited locomotion, and arched threat displays toward other cats, as the result of inhibited pouncing. [1]

Thus, the learning problem that the cat is engaged in changes as it develops. At first cats seem to be mostly interested in learning how to execute the individual motor patterns, then the entire behavior, then as they grow they become more and more interested in when to perform the behavior and how to integrate it with other important behaviors and goals.

There are a number of major trends that accompany this developmental timeline. Along with the increasing coherence and autonomy of the behavior system as a whole, the behavior becomes increasingly directed towards functionally useful stimuli. Instead of pouncing only on their littermates, as cats grow they pounce more and more on potential food sources such as mice and hamsters. Increasingly, the behavior also comes under the control of specific drives such as hunger. While young cats seem to hunt mostly as play, adult cats are more likely to hunt when they are hungry. The interesting thing about these trends is that they are elegantly matched to the animal's changing physical and perceptual abilities, to the changing demands of its environment as it gets weaned off of its mother's milk and protection, and to the changing learning tasks that the animal is engaged in. [17]

This developmental timeline is remarkably consistent with Kruijt's observations of the development of aggressive behavior in junglefowl [15]. While cats and junglefowl are very different species, and while predation and social aggression are very different behaviors, nevertheless a strikingly similar sequence of learning problems is confronted in both cases. Aggressive behavior in junglefowl starts to develop during the second postnatal week. At first, the component behaviors, locomotion, attraction, and pecking, appear individually and seemingly at random. Next, their form and orientation improve. Fighting gradually becomes more frequent and more complete. Aggression becomes more and more an autonomous system, and increasingly becomes under the control of external stimuli, which at first control the orientation of the behavior and then control both its initiation and orientation.

Finally, aggressive behavior becomes better integrated with the animal's other behavior, in particular escape behavior, which develops earlier. At first escape quickly inhibits fighting, but soon escape and fighting start to occur in rapid alternation. Later, escape and fighting can be activated simultaneously, and interactions between the two systems develop, with elements of escape being incorporated into fighting behavior. [15]

This gradual "phasing in" of aggressive behavior is mirrored later in the animal's life by the appearance of sexual behavior. Sexual behavior is at first infrequent and incomplete, easily suppressed by either escape or aggression. Then it starts to become more frequent and more complete, leading to conflicts with the earlier behaviors. Finally, all three systems can be active simultaneously, and new motor patterns seem to emerge from the interactions between the systems. [15]

The developmental timelines and trends outlined thus far also echo Fentress and McLeod's general observations about motor development [9]. They observe that the form of a movement often precedes its function. A given motor pattern will often be well coordinated before it serves any obvious functional purpose. They cite among other examples the preening movements in passerine birds, which appear before the feathers develop, as well as the gnawing and splitting movement used by squirrels to open nuts and the movements for dehusking seeds used by finches. They further observe that proper orientation, timing, and functional utility develop after motor coordination. Practice, it seems, is much less important for production than for effective use. Basic motor components are typically thought to have a strong genetic base, taking on an essentially invariant form under diverse developmental conditions, whereas the orientation, sequencing, and integration of these components are thought to be more dependent on experience.

2.2.3 Targeted motivation and reward.

Targeted motivation and reward signals, that are matched to the specific learning task that the creature is engaged in, can greatly improve the learning process. As cats develop, the learning tasks that they are engaged in change. A critical part of the success of this strategy is that the motivation and reward signals that they seem to be attending to also change. This fits in with our previous discussion of sensitive periods: specific learning episodes are often accompanied by increased sensitivity to specific types of information and environmental stimuli. Behaviors are motivated and expressed not only when they have direct functional utility, but also when their expression provides an opportunity for gaining information important for learning.

As discussed earlier, as behavior systems mature they often come under the control of specific drives such as hunger. Prior to this point, such behaviors are often largely self-motivated, and often are expressed in contexts where the costs of failure are low. Early aggressive behavior in cats is directed almost exclusively towards littermates. While such behavior is of little functional value, the learning that occurs in this context can be reapplied when the cat begins to hunt, providing significant advantage for the development of predatory behavior. Local reward signals associated with the individual predatory motor patterns must mediate this learning in the absence of the functional reward of eating. [17]

Leyhausen proposes that the important components of the cat predatory sequence are self-motivating [17]. Cats seem to be independently motivated to pounce and throw and chase, just as they are motivated to eat. They can often be observed to "play" with their prey for long periods of time before proceeding to eat them. Many explanations for this behavior have been proposed. One possibility stems from the observation that killing is rare in real situations. In the wild, much more time needs to be spent stalking and chasing than eating, since prey may be scarce and the kill itself may often fail. The levels of motivation for the individual predatory behaviors may be tuned to match the demand for their expression in the most frequently encountered natural situation. Since the individual behaviors are independently motivated, when presented with an easy opportunity to express them the cat may do so for quite some time, satisfying these independent motivations even though in doing so it delays killing and eating the prey. One consequence of this analysis is that it predicts that a cat that is simply presented with a mouse will spend more time "playing" with it than a cat that must hunt the mouse for a significant period of time before acquiring it. I am not aware of evidence that supports or contradicts this hypothesis; it would be interesting to pursue this line of inquiry further at a later point.

Local motivation and reward signals may also help to appropriately direct the consequences of failure. If chasing is rewarding in and of itself, then the cat will not get frustrated with this element of its behavior if it fails to kill its meal. If the cat fails to make the kill, it is the fault of the particular killing technique and not the entire predatory sequence. Independently motivated "play" behaviors also offer indirect advantages for predation, such as tiring out the prey animal and preparing the conditions for the kills. They may also benefit the animal through experimentation, providing new contexts for executing the final elements of the predatory sequence.

Hogan provides some interesting insights into the idea of targeted motivation and the phenomenon of prefunctionality in his discussion of pecking behavior in chicks [11]. In chickens, pecking behavior first appears prefunctionally, while the chick is still living off of the energy provided by the yolk. Pecking is initially not correlated with nutritional state, and thus motivated independently from hunger. During this prefunctional period, chicks learn to peck at things they can swallow, and can develop taste preferences. Eventually pecking and swallowing become associated with nutrition, and chicks begin to peck more when they are hungry.

Hogan points out that in most species, the primary source of nutrition and the method for obtaining it changes dramatically at least one or two times during the life of the animal. In mammals, the source of nutrition typically changes from the placenta to suckling to eating; in birds, it typically changes from the yolk to begging for food to foraging. Hogan argues that the timing of these transitions may be hard to specify ahead of time. For example, babies may be born late or premature, or the supply of breast milk may get interrupted. Experience derived from prefunctional behavior may provide the best timing for the behavioral shifts that must accompany these transitions in food source. [11]

Further, prefunctional behavior provides important experiential and exploratory advantage. If pecking were controlled from the outset by nutritional needs, the chick would only start to peck when it got hungry for the first time, after the resources provided by the yolk were exhausted. In this case the chick would need to execute the pecking behavior correctly right away. Pecking ahead of the need provides the chick with the opportunity to learn to peck correctly while the stakes are still low. [11]

2.3 Short-Term Engineering Goals and Experiments

In the short term, I believe that there are a number of ways that the lessons from animal development introduced above can be directly applied to the design of synthetic characters. In this section, I describe three ideas for integrating these lessons and evaluating their effectiveness for the learning process. I believe that these ideas represent some of the "low-hanging fruit" for this area, in that they are fairly easy to implement but may provide fairly significant advantages.

2.3.1 Learn one thing at a time.

Instead of trying to solve a large learning problem all at once, create a system of learning specialists that learns one major subproblem at a time, or perhaps two overlapping subproblems. Transitions between learning tasks could be triggered by a timer or by contextual cues, with the ultimate goal being to try to learn when the environment is best suited to teach. In order to evaluate the validity of this approach, it would be useful to perform a comparative analysis of its performance in a fairly constrained, formal learning environment such as a standard Q-learning environment. Ideally, this analysis would demonstrate that this approach performs better than the unstructured approach, in that the system learns more quickly or receives reward more consistently during the learning process. My hypothesis is that such a system would trade off on asymptotic performance somewhat for a significant gain in learning speed.

2.3.2 Perceptual templates.

Perceptual templates associated as releasers for specific behavior systems or for individual behaviors within the systems could seed and help direct the learning process. The template for hunting behavior, for example, might originally be "small, moving" but might then be refined through experience and behavioral feedback into a complex of features representing a mouse, or into a number of different, specific templates. This might be particularly interesting because it would allow us to start exploring some ideas about long-term memory, modifying a persistent, long-term template in response to salient features of the immediate perceptual environment. My hypothesis is that this approach could be readily shown to result in better learning performance than systems that start out with no prior perceptual knowledge. Perceptual templates could help greatly reduce the size of the perceptual space that needs to be explored during the learning process, and thus represent a good way of leveraging small amounts of prior domain knowledge for significant advantage.

2.3.3 Between-system coherence: scalable system integration.

As discussed previously, the last thing that often seems to get learned in development is coherence and integration between different behavior systems. The predatory system as a whole gets "tuned in" to a complete behavioral repertoire that includes escape; later, sexual behavior gets tuned in to a complete behavioral repertoire that includes predation and escape. It would be great to demonstrate a behavioral framework that could support bringing in new behavior systems without destabilizing the entire system, i.e. without resulting in dithering or obsessed behavior. This behavioral tuning might be context-specific: cats seem to learn the right balance of aggression and escape to direct towards their littermates, and the right balance of predation and escape to direct towards different prey animals. Perhaps one of the things that gets learned about specific objects or types of objects is what high-level balance between behavior systems to apply when directing behavior towards those objects. These ideas represent interesting lines of inquiry which could be readily explored within a computational framework.

In the next chapter, I describe a new learning algorithm that stems from these ideas, in particular the first and third idea discussed above. This new algorithm is implemented as an extension to a popular formal technique for learning in synthetic agents called Q-learning.

Chapter 3

Hierarchical Q-Learning

This chapter introduces timed hierarchical Q-learning, a new learning algorithm that incorporates some of the lessons from animal development discussed in the previous chapter. In presenting this new algorithm, I first describe the basic core learning algorithm, Q-learning, and review some related research into hierarchical Q-learning.

3.1 Q-Learning Overview

Q-learning is a popular technique for solving a broad class of tasks known as reinforcement learning problems. In reinforcement learning, an agent interacts with a dynamic environment which provides the agent with feedback about its behavior. The agent's job is to modify its behavior in such a way that it generally increases the amount of positive feedback that it receives from the environment.

Reinforcement learning approximates the world by dividing it up into discrete sets: the environment, time, and the agent's actions and observations are all represented discretely. At each time step, the environment is in a particular state, s. The agent receives as input an observation, o, that represents its perception of this current state. Based on its observation, the agent chooses an action, a, to perform. The action changes the state of the environment, and based on this change the agent receives the scalar reward value r. s can be any one of a discrete set of states S, o any one of a discrete set of observations O, and a any one of a discrete set of actions A. r can be any scalar value, though it is typically between 0 and 1.

In many simple reinforcement learning problems, it is assumed that the agent can directly perceive the exact unique state of the environment, s, i.e. that there is a direct one-to-one mapping between *S* and *O*. We will follow this convention in introducing basic Q-learning, and use *s* in place of *o* in the following discussion.

The agent's goal is to modify its own behavior so as to maximize some measure of the reward that it receives over time. While there are many ways to measure reward received over time, the most popular and theoretically appealing method is called infinite-horizon discounted reward. In this method, reward values are simply summed over time, but values farther into the future are more and more heavily discounted by a temporal discount rate, γ. Thus at any given time step *t*, the agent wants to choose the action that maximizes the expected value

$$E\left(\sum_{n=0}^{\infty} \gamma^n r_{t+n}\right) \tag{3.1}$$

where r_{t+n} is the reward received *n* time steps into the future. The closer γ is to 1, the more the agent will value rewards received in the distant future, and the closer γ is to 0, the more the agent will chose actions that result in immediate reward.

The agent seeks to find an optimal policy for its behavior, which will maximize this expected value of reward at every time step. In the simplest case, this policy is represented as a function, π, mapping states to actions: $\pi(s)$ evaluates to the best action to take in state *s*.

Q-learning is an algorithm based on dynamic programming that attempts to learn the optimal policy by building a table that contains a value for every possible pairing of state and action. The values in the table are known as Q-values. A particular Q-value, denoted as $Q(s, a)$, represents the expected value of taking action *a* in state *s*, and then following the optimal policy from then on.

Once all of the Q-values are known, the optimal policy becomes easy to describe. In a given state *s*, simply choose the action *a* with the largest associated Q-value.

$$\pi(s) = \text{argmax}_a Q(s, a) \tag{3.2}$$

Similarly, the expected value of arriving in state *s* is simply the value of the largest Q-value associated with *s*.

$$E(s) = \max_a Q(s, a) \tag{3.3}$$

Of course, when the agent starts out it has no way of knowing what the right Q-values are for representing the optimal policy, and must by necessity start out with arbitrary Q-values. These initial Q-values are then revised with experience to be better and better approximations of the actual Q-values. At each step, the agent takes some action *a* in state *s*, receiving reward *r* and causing a transition into state *s'*. It then updates Q-value $Q(s, a)$ according to the following simple rule:

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a') \tag{3.4}$$

The Q-value is thus updated to be the sum of the immediate reward and the discounted expected value of arriving in state *s'*. In practice, a learning rate, α, is often incorporated into this update equation, making the update more robust to noise:

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \max_{a'} Q(s', a')) \tag{3.5}$$

With every iteration, a small amount of real information from the environment gets incorporated into the Q-values. It makes some intuitive sense, then, that the Q-values will become better approximations of the actual Q-values as time passes. What makes Q-learning so interesting theoretically is that it can be proven that the Q-values will always converge to their actual values, given a few assumptions about the behavior of the environment, and as long as every state-action pair gets explored often enough. With enough exploration, the Q-values will always converge to a representation of the optimal policy.

Another important point is that convergence will occur regardless of the action selection strategy that the agent uses during learning, as long as it engages in enough exploration. This leaves open the interesting question of how best to balance exploitation with exploration. When the Q-values are close to convergence, the best thing for the agent to do is to greedily choose the action with the largest Q-value for the current state. However, if the agent does not explore enough early on, the Q-values will be slow to converge. Many different techniques for balancing exploration and exploitation have been proposed, but one common strategy is to choose actions greedily most of the time, but with some small probability choose a random action.

While Q-learning is a powerful and theoretically attractive learning algorithm, it does suffer from a number of drawbacks. Convergence can be very slow, often requiring every state-action pair to be visited numerous times. State and action spaces can grow very large and can sometimes be continuous, greatly increasing the time and memory requirements of the algorithm. In general, Q-learning scales too poorly to accommodate the size of state space commonly encountered in practical applications. Many researchers have tried to confront this problem by developing techniques for generalizing learning across similar states or similar actions. Such generalization attempts to take advantage of the rich structure inherent in many state and action spaces. Neural networks are often used as a tool for making these generalizations [16].

Q-learning also runs into problems when the mapping between environment states and agent observations is no longer one-to-one. This typically occurs when certain states of the environment are hidden from the agent because of its limited perceptual abilities. If the agent can not uniquely identify the state of the environment, the so-called optimal policy may actually perform quite badly. To overcome this problem, the agent must use memory or other types of internal state to disambiguate important environmental states. A very effective example of using memory in this way is described in [18].

For a more in-depth discussion of Q-learning, its drawbacks, and extensions, please see [14].

3.2 Hierarchical Q-Learning - Related Work

Hierarchical Q-learning describes a loosely related collection of algorithms derived from Q-learning, all of which attempt to improve upon the original by hierarchically restructuring the action selection decision in some way. Hierarchical Q-learning systems typically consist of two important pieces: leaf controllers, and a selection policy. The leaf controllers each try to solve some subproblem of the agent's full task, and in many cases each learns its own action policy via standard Q-learning. The selection policy tries to choose which leaf controller should be in charge of the agent's actions at the current moment, and may engage in learning as well. In some cases there may be multiple levels of selection policies, with each selection policy trying to decide which sub-selection policy should be active.

Such hierarchical systems can be quite advantageous for learning, since it is often easier to learn each of the subtasks and how to coordinate them than to learn how to solve the full, unstructured problem. It should be noted that in some cases, decomposing the full problem hierarchically may mean that the optimal policy can no longer be represented. However, many of these hierarchical learning systems are designed with the idea that fast and pretty good is often preferable to slow and optimal.

One way to hierarchically divide up a problem is by segmenting the input space, essentially creating multiple different agents, each specializing in a particular region of the state space. Dayan and Hinton describe an algorithm called feudal Q-learning which constructs a hierarchy of agents operating on progressively finer resolution views of the state space in a maze solving task [7]. High level agents operate on a very low resolution view of the state space, issuing gross navigational commands which lower level agents learn to satisfy. Using this technique, they demonstrate learning which is slow initially but faster overall as compared with standard Q-learning.

Kaelbling describes a related technique called landmark learning [13]. An environment for a navigation task is seeded with a number of landmarks, which essentially provide a lower resolution view of the space. The agent navigates via the landmarks until it is close to the landmark closest to the goal, at which point it navigates for the goal directly. This is an interesting approach which also brings up an interesting trade-off related to the landmark density: with more landmarks, the high level search gets harder, but the low level search is easier and the paths followed are more direct. Conversely, with less landmarks, the high level search is easier but the low level search becomes harder.

Sun and Peterson introduce an algorithm which is capable of automatically discovering a useful decomposition of the state space [21]. Their region splitting algorithm assigns one agent to each region of the state space, trains each agent, and then conducts offline optimization to improve its partitioning of the state space. This process is repeated until it stabilizes with a given number of disjoint regions, each with its own specialized agent. During the optimization step, the goal is to increase the consistency of the predictions made by the agents about the rewards they receive, thus hopefully making the environment more predictable by the agents. This technique is shown to result in good performance on a complicated maze solving task.

While these techniques for simplifying Q-learning problems by dividing up the state space are interesting and fairly effective, my work has focused on speeding up learning by dividing the full problem up into subtasks, identifying subgoals and learning strategies for solving them. My exploration of hierarchical Q-learning stems largely from the work of Humphrys.

Humphrys describes a large, multi-goal "house robot" problem that is made simpler through hierarchical action selection [12]. Instead of learning one large Q-table, his system learns a number of sub-policies, each with their own reduced state space and simple reward function, and at the same time learns a selection policy that operates on the full state space, selecting a particular sub-policy at every time step. This task decomposition significantly improves the learning performance.

Humphrys' work also explores the idea that instead of learning the selection policy, a fixed selection policy may also work very well in certain circumstances. In his approach, sub-policies negotiate for control based on their Q-values, using different arbitration heuristics. One heuristic, for example, tries to minimize the worst unhappiness amongst the sub-policies, while another tries to maximize the best happiness. One drawback of these techniques is that since the sub-policies negotiate based on the Q-values, a critical emphasis is placed on the actual values used in the local reward functions. Humphrys optimizes these values using a slow genetic algorithm based search, though once optimized, these values result in excellent on-line learning performance.

There are a number of ways that these approaches might be improved using ideas from development about the temporal organization of learning. For one, the learning might proceed more quickly if the individual sub-policies were learned one at a time, followed by learning of the selection policy. In the case of the fixed selection strategies, instead of using GA-based search to weight the sub-policies, the sub-policies could be "tuned in" one at a time. This process could be regulated by the global reward function, perhaps in conjunction with some measure of the level of dithering or obsessiveness in the resulting behavior. The first of these ideas will be discussed in greater length in the following section, while the second is left as future work.

Humphrys' work makes the important point that learning can be significantly improved by hierarchically breaking up the problem into selection policies and sub-policies. This point is also made by the work of Parr and Russell, who present a language for designing hierarchical finite-state machine controllers for reinforcement learning agents [19]. In their framework, the states, actions, and transitions which define the controllers can be specified partially, allowing for refinement via Q-learning. This allows the designer to efficiently introduce prior knowledge about the structure of the problem, which can greatly accelerate the learning process. In a similar vein, Dietterich describes an approach called MAXQ decomposition that allows the designer to specify a directed acyclic graph of tasks and subtasks [8]. This speeds up learning, particularly since the subtasks may be able to get away with ignoring large sections of the state space. Dietterich provides some useful formal results about when such state-space abstractions are safe to perform.

An alternative approach is presented by the work of Sun and Sessions, who discuss a technique for automatically discovering useful subtasks for dividing up the global task [22]. Their system consists of three basic components: simple action policies, termination policies paired to these action policies, and selection policies. These components can be layered into multilevel hierarchies, allowing for the automatic segmentation and discovery of important action sequences. This system is remarkably general, as it involves no prior knowledge specified by the designer, but it also suffers on these grounds in terms of the complexity of the problems that it is able to tackle in practice.

There are a number of particularly related works which echo some of the lessons from development about the temporal organization of learning. Singh discusses compositional learning, in which the system first learns elemental tasks, and then learns how to order these tasks to solve composite tasks [20]. Here the composite tasks are strict sequences of elemental tasks, and thus what gets learned is not a full selection policy but rather a gating function that sequences the elemental policies.

Singh also discusses two different strategies for training the composite tasks. The first strategy is to learn all of the elemental tasks first followed by the composite task. The second strategy is to learn a succession of tasks, where each new task requires the previously introduced elemental tasks plus one new elemental task. Singh provides good empirical results for this "shaping" strategy, although he does not compare it against unstructured training.

Tham elaborates somewhat on Singh's compositional Q-learning for a complicated robot arm positioning task [23]. He allows for more general reward functions for the elemental tasks, and permits the agent to have multiple independently controlled actuators. Tham discusses the temporal structure of the learning process, mentioning that the elemental tasks needed to be learned before the composite tasks in order to achieve reliable results, but he does not quantify this point.

Of particular note is the work of Lin, who introduces a number of important ideas about structured learning [16]. Lin uses hierarchical learning to speed up learning of a battery charging task in a simulated mobile robot domain. He uses Q-learning to learn the elementary skills (wall following, door passing, and docking with the charger), and then uses Q-learning again to learn the selection policy. The agent learns the subtasks individually in special training sessions before learning the composite task. Lin discusses using direct teaching, guiding the agent step by step through a few examples of correct behavior, to accelerate learning and overcome local maxima. This teaching significantly improved the learning results.

Lin employs two interesting rules to constrain exploration during the learning of the selection policy: an "applicability" rule and a "persistence" rule. The applicability rule is designed to choose only applicable actions; in practice, by thresholding the Q-values provided by the subtasks. The goal of the persistence rule is to not switch skills unless some significant change occurs to the robot's situation. Here "significant change" is defined in a fairly ad hoc fashion: either the goal of an elementary skill gets achieved or a previously inapplicable skill becomes applicable.

Lin's empirical results compare unstructured learning with a few teaching examples against hierarchical learning with pre-trained subtask policies. The hierarchical learning takes just about as long but results in a policy with greatly superior performance, although the use of the persistence rule is a critical piece of this improved performance. Lin focuses on the value of hierarchical learning in general rather than on how the subtask policies get phased in, and does not empirically validate the training schedule used or the temporal dynamics of subtask incorporation. All in all, though, his work makes a strong case for temporally structured learning.

Kaelbling makes a few related points in her excellent review of reinforcement learning [14]. In her conclusion, she identifies the development of methods for incorporating prior bias into learning systems as critical for the future of reinforcement learning, and for scaling up from simple problems to hard ones. In particular, she discusses shaping, the idea of presenting the learner with simple problems before more complex problems, as one promising way of incorporating such bias.

3.3 Timed Hierarchical Q-Learning

How can the lessons from animal development discussed in the previous chapter be used to enhance hierarchical Q-learning? This section describes a number of potential extensions to standard hierarchical Q-learning which get wrapped together into a new algorithm called timed hierarchical Q-learning. Here, by standard hierarchical Q-learning, I refer to the approach wherein Q-learning is used to learn a selection policy for choosing amongst a fixed set of basic skills, where each of the basic skills is itself simultaneously learned using Q-learning.

3.3.1 Learn one thing at a time.

One simple way of extending standard hierarchical Q-learning is to constrain the learning rate used by the main selection policy so that the agent learns how to perform each skill before learning how to coordinate amongst the skills. By setting the selection policy's learning rate to zero while the skills are being learned, the agent avoids learning prematurely based on consequences which do not reflect accurate performance of the skills. In some cases, delaying selection policy learning may increase the time it takes to learn a solution to the overall task. In many cases, however, it may be very beneficial to avoid premature learning, particularly in situations where individual states are visited only rarely. In such situations, premature learning may be hard to overcome through later experience, and may be especially damaging if it causes the state's utility value to change incorrectly, thus threatening the Q-values associated with neighboring states.

3.3.2 Introduce one skill at a time.

In addition to constraining when learning occurs, an interesting technique for guiding what the agent learns is to introduce one skill at a time. In such an approach, the agent starts out with only one skill to choose, and then additional skills are added to its available repertoire as time passes. Of course, if the agent can only receive reward by using its entire set of skills, then this approach is just a waste of time. However, if the agent can receive some reward using only one or just a few of its skills, this approach may offer some advantages over starting the agent off with a full repertoire of skills.

Using this technique, the agent starts off with a single skill, choosing it over and over again. Thus reward value is initially propagated throughout the state space with respect to this simple strategy. When the second skill gets introduced, its value is initially learned in each state with respect to the existing strategy of choosing the first skill over and over again. In states where choosing the second skill gets the agent closer to reward than the first skill with respect to this existing strategy, the value of the second skill will grow to surpass that of the first skill. Similarly, when the third skill is introduced, its value is initially learned with respect to the strategy of choosing the better of the two previously introduced skills in each state.

As each new skill gets introduced, the agent learns the value of the new skill with respect to the best strategy available given its current skills, assuming sufficient learning has occurred. Thus the agent's job essentially becomes to determine whether or not the new skill represents an improvement over the existing strategy in each state. This is an interesting and potentially useful re-framing of the standard Q-learning problem.

This technique would seem to work best in cases where the skills introduced early on are capable of comprising a quick, approximate solution to the problem at hand, while skills introduced later on provide increased sophistication, allowing the agent to receive reward more quickly or from a wider range of states. Thus this approach is similar to some of the "shaping" techniques discussed in the previous section, where simple but successful strategies are gradually elaborated on as the agent develops.

3.3.3 Motivate each new skill to encourage exploration.

When using the technique of introducing skills one at a time, standard random exploration may not be sufficient for rapidly incorporating new skills into the agent's existing strategy. One potential improvement is to motivate the agent to experiment with each new skill vigorously for a short period of time following its introduction. While this would temporarily interrupt the existing strategy, it might significantly speed up the learning process by forcing the agent to try out the new skill in a wide range of states.

3.3.4 Description: timed hierarchical Q-learning.

Timed hierarchical Q-learning wraps up all three of the heuristics described above into a single algorithm that represents a fairly straightforward extension to standard hierarchical Q-learning. Two major components differentiate timed hierarchical Q-learning from the standard algorithm: skill timers and biased exploration episodes.

A single skill timer must be specified for each skill to be added to the agent's repertoire. Each skill timer consists of two values: a time stamp, which specifies when the associated skill should be introduced, and a duration, which specifies for how long after the introduction of the skill the learning rate of the selection policy should be set to zero. By setting the learning rate of the selection policy to zero for a short period of time following the introduction of the new skill, premature learning can be avoided.

When a new skill gets added to the agent's repertoire, the Q-value for that skill in each state gets set either to the standard initial value or to the current utility value for the state, whichever is lower. This guarantees that adding a new skill will leave unchanged all of the existing state utility values, which are extremely valuable sources of information for the learning process.

The second major component used in timed hierarchical Q-learning is biased exploration episodes. Biased exploration episodes, like skill timers, are specified for particular skills, although there may be more than one specified for each skill. A single biased exploration episode is defined by three values: a time stamp, which specifies when the episode should start, a duration, which specifies the duration of the episode, and an exploration probability. The exploration probability specifies the probability with which the agent should select the associated skill during the exploration episode. When the associated skill is not selected, the agent should simply choose one of the other skills as directed by the standard selection policy.

Typically, biased exploration episodes occur as each new skill gets introduced, forcing the agent to experiment thoroughly with the new skill. The exploration probabilities specified for the episodes provide a useful way for the designer to incorporate a limited amount of prior domain information into the learning process. For example, the designer might set the exploration probability to be quite high if she knows that it is safe to perform the skill with high frequency, whereas the designer might set the probability significantly lower if she knows that performing the skill with high frequency might be dangerous or get the agent stuck.

Now that we have introduced the timed hierarchical Q-learning algorithm, we need to establish what it is useful for. The following chapter provides an evaluation of the algorithm, comparing its performance against that of standard Q-learning and hierarchical Q-learning in a simplified predator-prey domain.

Chapter 4

Evaluation and Results

In order to evaluate the performance of the timed hierarchical Q-learning algorithm, I constructed a simulation centered around a simplified predator-prey interaction. In the simulation, the predator's goal is to learn how to approach the prey without scaring it away. I conducted a number of experiments in this domain comparing the performance of timed hierarchical Q-learning against standard Q-learning and hierarchical Q-learning. This chapter describes the simulation environment and the experiments, and provides an analysis of the results obtained. Taken as a whole, the results support the conclusion that timed hierarchical Q-learning is a useful extension of standard hierarchical Q-learning.

4.1 Simulation Environment

The simulation environment consists of a grid world inhabited by a predator agent and a prey agent. The predator and prey each occupy one space on the 120-by-120 square grid. The predator's goal is to move into the space occupied by the prey, thereby consuming the prey.

The prey agent is always located at the center of the grid, and can face in any one of the eight cardinal directions. The prey does not move, except that if it notices the predator making progress towards it, it flees and escapes. The prey has a limited visual sensor that allows it to detect the predator. The prey's visual field is semicircular, extending out in front of the prey in the direction that it is facing to a radius of 27 units. Thus, in order to successfully capture the prey, the predator must flank around the prey and approach it from behind. If the predator starts in or accidentally wanders into the prey's visual field, it must back away to a distance where it can flank around the prey undetected.

4.1.1 The predator.

The predator has no orientation, and can move in any of eight directions or opt not to move. At every time step, the predator selects one of these nine possible actions, and as a result either moves into one of the adjacent grid spaces or stays where it is.

The predator has a number of sensors that provide it with information about the prey. It can sense the direction from itself to the prey as one of 24 discrete directions. It also can sense the distance from itself to the prey as one of 18 discrete distances, with greater resolution afforded to distances that are closer to the prey. In addition, the predator can detect in which of the 8 directions the prey is facing. Thus the predator's total state space consists of 24 x 18 x 8 = 3,456 distinct states. Combining that with the predator's available actions, there are a total of 9 x 3,456 = 31,104 distinct state-action pairs for the predator to explore and learn about.

The predator receives reward in only two situations. If it takes an action which moves it into the same space as the prey, it receives a reward of positive 100 points. On the other hand, if it takes an action that causes the prey to flee, it receives a punishment of negative 50 points. At every other time step, the predator receives 0 points of reward.

4.1.2 Trial setup.

A number of experiments were conducted in this domain as described in the following sections. Each experiment consisted of a specified number of discrete trials, with the predator continuing to learn across trials. At the beginning of each trial, the predator is placed at a random radius between 10 and 60 units away from the prey, and at a random angle on the circle defined by that radius. The orientation of the prey is also set randomly. Each trial runs for 500 time steps, or until the predator either captures the prey or causes it to flee. At the end of the trial, the total reward received - either 100, 0, or -50 points - is recorded, along with a time stamp equal to the total number of simulation steps since the beginning of the experiment.

The graphs of experiment results presented in the following sections plot this recorded data, charting reward versus time elapsed, where reward has been averaged across windows of consecutive trials. This averaged reward represents something akin to a success rate: the maximum averaged value is 100, representing 100 percent success, while the minimum averaged value is -50, representing 100 percent failure. Algorithms thus fare better on these graphs by converging more quickly to a successful strategy. Successful strategies, however, may differ in terms of how quickly they allow the predator to catch the prey, and this information is not reflected in the graphs. Since catching the prey quickly is not particularly important in this domain, I contend that this is not much of a loss, and is a small price to pay to have clear upper and lower bounds on the charted data.

4.1.3 Why this is a difficult problem.

31,000 state-action pairs may seem like a large space to explore, but it is in fact quite modest in comparison to many other reinforcement learning domains. What makes the predator's problem difficult is that reward is very sparse in this domain. Since the predator only receives nonzero reward if it captures the prey or scares it away, in some cases the predator may go for hundreds of time steps without receiving any feedback. This can make learning very difficult, and indeed, in many of the experiments the algorithms under consideration required something on the order of 3 million time steps to converge to successful strategies. This slow learning process allowed for significant differences to emerge between the various algorithms.

4.2 Algorithms Tested

The experiments that I conducted involved examples of all three of the major learning algorithms discussed in the previous chapter: standard Q-learning, hierarchical Q-learning, and timed hierarchical Q-learning. This section describes how each of these algorithms was set up for the experiments and adapted for the predator-prey problem domain.

4.2.1 Standard Q-learning.

The standard Q-learning algorithm used in the experiments involved a single Q-table operating on the predator's full state-action space. Thus in this approach the predator tried to learn which of its nine actions was best to perform in each distinguishable state. The predator used a standard greedy-with-randomness action selection policy, selecting the action with the highest Q-value most of the time, but with some probability selecting a random action. This random selection probability was set at 0.1 for all experiments. Also for all experiments, the standard Q-learning algorithm used a temporal discount factor of 0.9 and a learning rate of 0.6.

4.2.2 Hierarchical Q-learning.

The hierarchical Q-learning algorithms used in the experiments divided up the problem using as many as three different subskills: approach, circle clockwise, and avoid. Simple local reward metrics associated with each subskill guided the learning of that skill. The approach skill received reward when it caused the predator to make progress towards the prey, the circle clockwise skill when it caused the predator to make progress around the prey clockwise, and avoid when the predator made progress away from the prey. Standard Q-learning was employed to learn each of the subskills as above. Each subskill could select any of the predator's nine basic actions, but operated on a smaller subset of the predator's full state space. The subskills ignored the prey's orientation, and thus derived state information from only the direction and distance between the predator and the prey. Since the subskills each operated on a reduced state space and received reward frequently, they could be learned individually relatively quickly, often achieving good performance within 50,000 time steps.

In addition to learning the subskills, the hierarchical Q-learning algorithms simultaneously used standard Q-learning to attempt to learn which skill to select in each state in the predator's full state space. Settings for learning this Q-table were as described above, except that the learning rate was set significantly lower - to 0.1 - in order to accommodate the randomness and noise inherent in the learning processes associated with the subskills.

4.2.3 Timed hierarchical Q-learning.

The timed hierarchical Q-learning algorithms used in the experiments were set up identically to the standard hierarchical Q-learning algorithms described above, using the same subskills, state spaces, and learning parameters. These algorithms of course employed their own timing heuristics and exploration biases as described in the previous chapter, but otherwise were as similar as possible to their standard hierarchical counterparts.

4.3 Experiments and Results

4.3.1 Performance in standard environment.

The first experiment that I conducted was designed to compare the performance of the three major learning algorithms in the standard environment described above. In all, four different algorithms were tested: standard Q-learning, hierarchical Q-learning with the approach, circle clockwise, and avoid skills, timed hierarchical Q-learning with the same three skills, and finally hierarchical Q-learning with just the approach and circle clockwise skills. Each algorithm was run three times in the simulation, and its results were averaged together. Standard Q-learning was simulated for 24,000 trials per run, 3-skill hierarchical Q-learning for 24,000 trials, and 2-skill hierarchical Q-learning for 48,000 trials.

Timed hierarchical Q-learning was simulated for 48,000 trials. Its action timers were specified such that the predator started out with only the approach skill, and then added the circle clockwise skill on trial 12,000 and the avoid skill on trial 14,500. After each new skill was introduced, the learning rate for the main selection policy was set to zero for 300 trials in order to give the predator time to learn how to perform each skill. The biased exploration episodes were specified such that the predator experimented frequently with each new skill for 2000 trials following its introduction, selecting the circle clockwise skill with 95 percent probability and the avoid skill with 30 percent probability. The probability for the avoid skill was set lower to prevent the predator from drifting away and spending most of its time on the periphery of the environment.

The results of this experiment are displayed in the following graph, which charts averaged performance versus simulation steps elapsed.

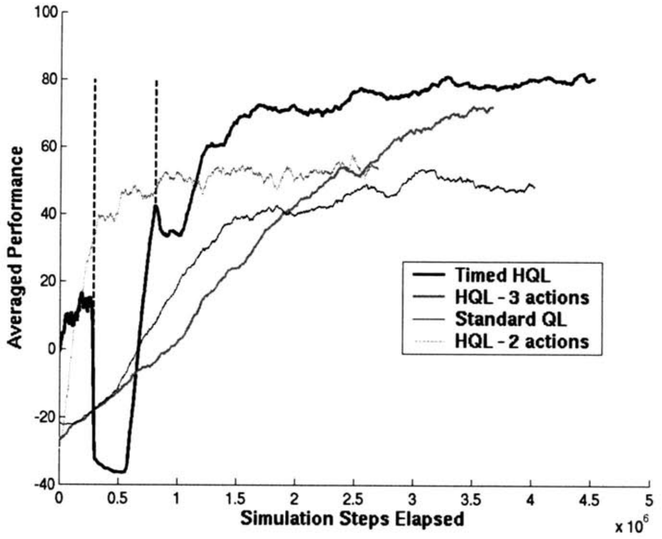

Figure 4.1: Performance in standard environment. Dashed lines mark the introduction of first the circle clockwise skill and later the avoid skill.

Timed hierarchical Q-learning (thick dark line) significantly outperforms both hierarchical Q-learning (thick light line) and standard Q-learning (thin dark line) in this domain, converging a good deal faster than either of these two algorithms to an effective strategy. Hierarchical Q-learning with only two skills (thin light line) converges very quickly, but achieves lower performance at convergence because the predator does not have the ability to avoid the prey if it starts in or randomly wanders into the prey's visual field.

Both hierarchical Q-learning and standard Q-learning, which start the predator out with a full set of available actions, must essentially rely on a random walk of the state space during the early phase of the learning process. The predator must randomly walk its way into the prey in order for any reward to be received, and this process must be repeated over and over again early on in order to start pushing reward value throughout the state space. In contrast, hierarchical Q-learning with just two skills constrains the predator to choosing only actions which move it on average closer to the prey. This approach thus receives reward much more consistently early on, and consequently learns much more quickly.

Timed hierarchical Q-learning can be thought of as an attempt to take advantage of the rapid early learning of 2-skill hierarchical Q-learning, while still holding out the potential for good performance at convergence. Two significant dips in performance can be seen on the thick dark line, which correspond to the periods of increased experimentation which directly follow the introduction of the circle clockwise and avoid skills (marked by the dashed lines). By experimenting early on mostly with the approach skill and then the circle clockwise skill, the predator rapidly spreads reward value throughout the state space. The state utility values of individual states rapidly take on good approximate values. These approximate utility values are critical for rapidly incorporating the avoid skill into the predator's strategy following its introduction.

The spreading of positive reward value throughout the state space during the different stages of timed hierarchical Q-learning can be understood in fairly graphical terms. Initially, while the predator can only select the approach skill, positive reward value gets propagated back and away from the prey in a cone, opposite the direction in which the prey is facing. When the circle clockwise skill gets introduced and the predator starts experimenting with this new skill, reward value gets propagated away from this cone behind the prey, spreading in a counter-clockwise direction throughout a ring around the prey. Finally, when the avoid skill gets introduced, reward value gets propagated back in from this ring towards the prey and into its field of vision. When the exploration episode for the avoid skill ends, the algorithm finally starts to select actions based on the standard policy, and the predator rapidly starts to perform quite well. In this domain, where reward is rare, this strategy of constrained exploration and value propagation offers significant advantages over the random walk strategy employed by the other two algorithms.

4.3.2 Scaling with environment size.

The second experiment I conducted attempted to further analyze the performance of the timed hierarchical Q-learning algorithm by varying the size of the simulation environment. In addition to the standard 120 by 120 grid environment, the algorithms were tested in 60 by 60 and 150 by 150 grid environments. The other important distances used in the simulation, such as the prey's radius of vision, were scaled along with the size of the environment. Increasing the size of the environment makes the learning task more difficult, because it increases the average initial distance between the predator and the prey and also increases the total area for the predator to explore.

The same algorithms were tested in this experiment as in the first experiment, and the same timing and exploration parameters were used for the timed hierarchical Q-learning algorithm. The results of this experiment are displayed in the following three graphs, which were generated using the 60 by 60 environment, 120 by 120 environment, and 150 by 150 environment, respectively.

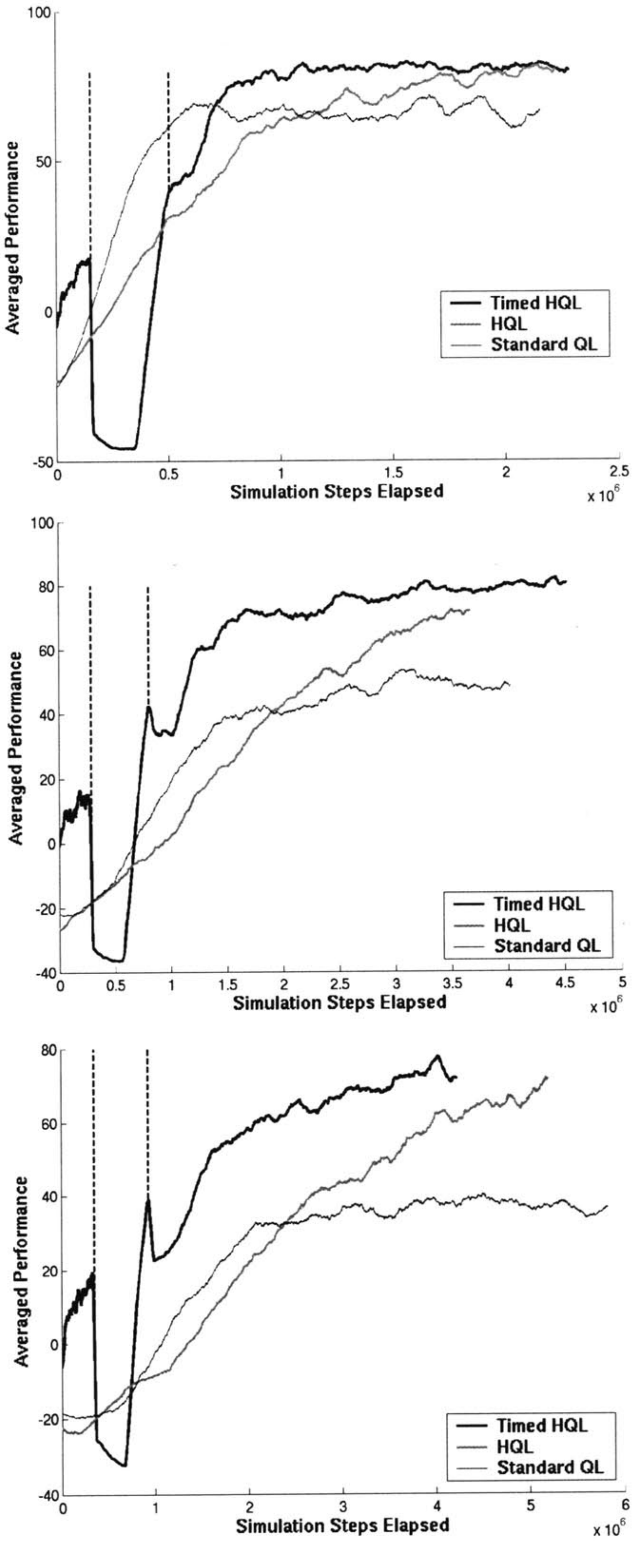

Figure 4.2: Scaling with environment size: 60 by 60, 120 by 120, and 150 by 150 environment grids. Dashed lines mark addition of circle skill and then avoid skill.

In the 60 by 60 environment, the differences between the three algorithms are fairly minimal. The timing and exploration heuristics used by the timed hierarchical Q-learning approach do not seem to offer much of an advantage in this domain, and while it does converge slightly faster than hierarchical Q-learning, it performs worse than standard Q-learning in this environment, at least during the early stages of the learning process.

The differences between the algorithms become much more pronounced in the larger environments. All three algorithms run more slowly, although the timed hierarchical Q-learning algorithm continues to converge to a good strategy in a relatively short period of time. The performance difference between the timed hierarchical Q-learning algorithm and the other two techniques increases significantly in the jump from the 120 by 120 environment to the 150 by 150 environment. This increase in the advantage of the timed hierarchical Q-learning algorithm is most likely a result of the fact that increasing the size of the environment reduces the frequency with which the algorithms that rely on a random walk of the state space receive reward early on, whereas the constrained exploration used by the timed hierarchical Q-learning algorithm continues to provide reward fairly consistently.

4.3.3 Scaling with state space resolution.

In my third and final experiment, I varied the resolution of the state space used by the main selection policy for each of the algorithms in the standard 120 by 120 grid environment. I changed the resolution of both the predator's direction to prey sensor and its distance to prey sensor. In addition to the high resolution used in the standard environment of 24 direction states and 18 distance states, the algorithms were also tested using a medium resolution of 16 direction states and 11 distance states, and using a low resolution of 8 direction states and 6 distance states.

Decreasing the state space resolution might at first seem to make the learning problem easier, since it decreases the total number of states that need to be explored, but it actually makes the problem much more difficult, since the predator's information about its location relative to the prey becomes more ambiguous, thus making a coherent strategy more difficult to discover. In addition, decreasing the state space resolution increases the size of each individual state, which increases the frequency of transitions that start in a state and end in the same state as a result of the agent's action. These self-transitions can wash out important distinctions between actions and significantly slow down the learning process.

The same algorithms were again used in this experiment as were used in the previous experiments, and again the same timing and exploration parameters were used for the timed hierarchical Q-learning algorithm. The results of this experiment are displayed in the following three graphs, which were generated using the standard high resolution state space, the medium resolution state space, and the low resolution state space, respectively.

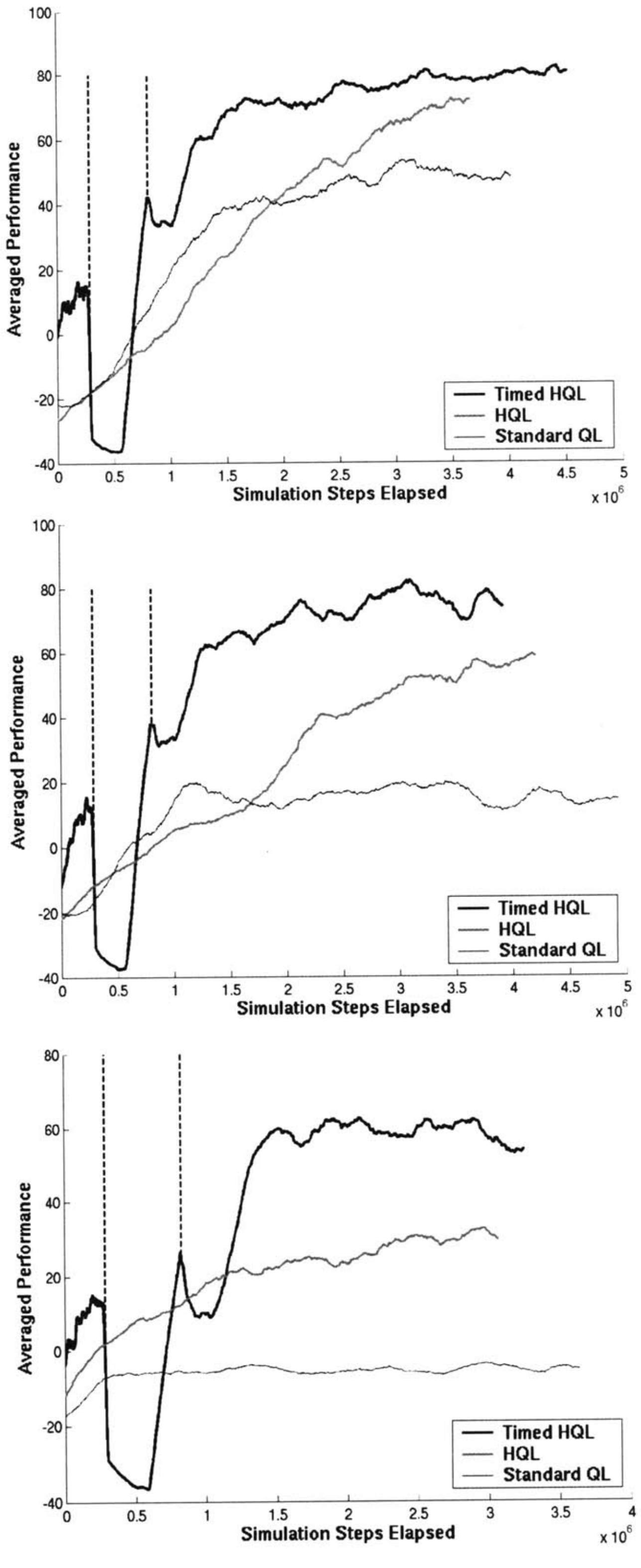

Figure 4.3: Scaling with state resolution: 24 direction states by 18 distance states, 16 by 11, and 8 by 6. Dashed lines mark addition of circle skill and then avoid skill.

As can be seen from the graphs, the standard Q-learning algorithm ceases to perform acceptably with decreased state space resolution, as it no longer has accurate enough state information to represent the optimal policy. The hierarchical, skill-based techniques, on the other hand, can still represent this best strategy. The performance of the standard hierarchical Q-learning algorithm, however, decreases significantly as the state space resolution decreases, while the timed hierarchical Q-learning algorithm continues to perform fairly well. Constrained exploration seems to be particularly useful in low resolution state spaces, as it reduces the number of within-state transitions during the important early phases of the learning process and forces important distinctions between action Q-values to increase to sustainable margins.

Taken as a whole, these three experiments support the conclusion that timed hierarchical Q-learning represents a useful extension of existing learning techniques. It provides good overall performance in this domain, and scales favorably compared to standard hierarchical Q-learning and standard Q-learning with respect to increasing environment size and decreasing state space resolution.

One remaining issue is that of the timing parameters used in the simulations of timed hierarchical Q-learning. The same timing parameters were used throughout the experiments. Since the environment varied significantly within and between the three experiments, the fact that the algorithm continued to perform well supports the idea that timed hierarchical Q-learning is fairly robust to the particular timing parameters chosen. However, this idea was not empirically tested, and it may be the case that the algorithm is fairly sensitive to the choice of timing parameters in some domains.

For these experiments, the timing parameters governing the introduction of new skills were chosen based on a limited amount of experimentation and some estimation about the speed of reward value propagation in this domain. For example, when the predator can only choose the approach skill, it can only move directly towards the prey, and thus reward value can only be propagated to approximately one new state per trial. Thus the predator must at a minimum participate in about 4,000 trials in order to propagate some reward value into every reachable state. Factoring in for randomness, and to achieve more solid state utility values, I chose 12,000 trials as a reasonable amount of exploration using only the approach skill. Similarly, I chose how many trials to allow the predator to experiment with just the approach and circle skills by estimating the number of complete circles around the prey that would have to be performed in order to propagate positive reward value away from the cone behind the prey and throughout the state space.

Choosing good timing parameters might be significantly more difficult in some domains, although hopefully it would be fairly easy in many cases. Ideally, it might be possible to automatically determine when to introduce new skills, making the designer's specification task significantly simpler and potentially speeding up the learning process. Good timing parameters allow approximate state utility values to be propagated to most of the states that are reachable given the agent's current set of skills. Thus one possible way to automatically determine when to introduce a new skill might be to measure the frequency with which the agent visits a state that it has not visited before with its current set of skills. When this frequency drops below some threshold, it might be a good time to introduce a new skill. Another possibility might be to introduce a new skill when the average change in Q-value per time step drops below a given threshold. Determining an effective metric for deciding when to introduce new skills presents an interesting opportunity for future work.

Chapter 5

Conclusion

This chapter offers some concluding remarks, providing a summary of the major contributions of this thesis and describing some directions for future work.

5.1 Summary of Contributions

I believe that the major contributions of this thesis are as follows. This thesis:

- Provides a targeted review of the literature on animal development from the perspective of designing systems that can quickly learn complicated and interesting behavior. Focusing on the development of predatory behavior in particular felid and canid species, this review identifies a number of design principles for the design of intelligent learning systems:

    1. First, the question when designing a learning system is often not so much how to learn, but what to learn and when.
    2. Second, learning skills in the right order, and one at a time instead of all at once, can often offer significant advantages to the learning system.
    3. Third, the inclusion of targeted motivation and reward signals that are matched to the specific learning task at hand can greatly improve the learning process.

- Introduces the timed hierarchical Q-learning algorithm, created through the application of the design principles enumerated above to standard hierarchical Q-learning, a popular approach to learning in autonomous agents. Timed hierarchical Q-learning extends standard hierarchical Q-learning by allowing the designer to specify action timers, which introduce new skills one at a time during the development of the agent, as well as biased exploration episodes, which specify how the agent should experiment with each new skill as it gets introduced.

- Provides a number of empirical results establishing the competitive performance of timed hierarchical Q-learning versus standard hierarchical Q-learning and standard Q-learning in a simulated predator-prey problem domain. Timed hierarchical Q-learning performs well in this domain, scaling favorably as compared to the other techniques with respect to both increasing environment size and decreasing state space resolution.

5.2 Future Work

Where do we go from here? As a first step, I think it is important to explore the ideas behind timed hierarchical Q-learning in a more general behavioral framework, such as that used in much of the other research conducted by the Synthetic Characters group. The Synthetic Characters behavior architecture allows for actions with temporal extent, and supports structured state spaces and agent memory, as well as a number of other improvements over standard Q-learning frameworks. While a Q-learning setting has been useful for the purposes of comparing timed learning with standard algorithms, future work in this area should take advantage of the flexibility and power of a more general behavior architecture, and ultimately be tested on more practical, real-world learning problems.

I contend that timed learning similar to that employed by timed hierarchical Q-learning will offer significant advantages in the context of the Synthetic Characters behavior system, even though this system differs in many respects from a Q-learning framework. The core point still applies: a random behavioral walk is not the best exploration strategy in many domains. Constrained exploration often works much better, and adding skills one at a time to the agent's repertoire can be an easy and effective means of constraining exploration. Timed skill introduction, in the best case, allows the agent to quickly learn a good approximate strategy, which can then be refined and improved upon as time passes. This approach should generalize well to a more flexible behavioral framework.

The ideas discussed in this thesis offer a number of interesting opportunities for additional work in the short term. First, as mentioned in the chapter on hierarchical Q-learning, with a rich enough set of skills it may be possible to use a static selection policy that chooses between the available skills based on the values in their Q-tables, instead of learning its own Q-table for selecting between the skills. Experimenting with different static skill selection heuristics might be an interesting area for future work.

Static selection heuristics may reduce the asymptotic performance of the algorithm somewhat, but they may also enable much faster learning. Such heuristics might be particularly effective if they were allowed to learn a set of weights for the available skills, essentially learning to prioritize the local reward signals used by the individual skills. This type of approach might be able to take advantage of timing and exploration heuristics similar to those used in timed hierarchical Q-learning. This would enable new skills to be "tuned in" one at a time to appropriate weighting values based on a global reward metric or some internal measure of behavioral coherence. A "tuning" approach, introducing one skill at a time, might achieve good performance while avoiding a potentially exponential search in the space of skill weights.

Another interesting area for future work might be to extend the timed hierarchical Q-learning algorithm by experimenting with opportunistic learning. As mentioned in the previous chapter, specifying when to introduce each new skill ahead of time may be a difficult task in some domains, and may not be appropriate in highly variable environments where the ideal timing parameters might differ significantly between runs. In such cases, it might be easier and potentially more powerful for the agent to use contextual cues to determine when to introduce each new skill. In a similar vein, it might be advantageous to use local context instead of prior timing to initiate exploration episodes. The agent could identify when it was in a good situation for experimenting with a particular action, either because that situation was specified ahead of time by the designer, or even because the agent itself learned about the types of situations that support fruitful experimentation. By taking advantage of these contextual cues, and potentially even engaging in this sort of reflective learning, the agent might considerably improve its learning performance.

Finally, I hope that the design principles and larger ideas about animal development presented at the beginning of this thesis will keep me and my colleagues in the Synthetic Characters group busy for quite some time. Animals are remarkably adept at learning complicated, coordinated behavior, and by paying close attention to the learning strategies that they use throughout their development, we may be able to make significant contributions to the field of intelligent systems design, and perhaps one day to the behavioral literature from which we draw our inspiration.

Bibliography

[1] Baerends-Van Roon, J. M., and G. P. Baerends. *The Morphogenesis of the Behavior of the Domestic Cat, with a Special Emphasis on the Development of Prey-Catching.* Amsterdam; New York: North-Holland Publishing Co., 1979.

[2] Bateson, Patrick, and Robert A. Hinde. "Developmental Changes in Sensitivity to Experience." In *The Development of Animal Behavior: A Reader*, edited by Johan J. Bolhuis and Jerry A. Hogan. Oxford: Blackwell Publishers, 1999.

[3] Blumberg, Bruce. "D-Learning: What Learning in Dogs Tells Us About Building Characters That Learn What They Ought to Learn." In *Exploring Artificial Intelligence in the New Millennium*, edited by Gerhard Lakemeyer and Bernhard Nebel: Morgan Kaufmann, 2002.

[4] Blumberg, Bruce, Marc Downie, Yuri Ivanov, Matt Berlin, Michael Patrick Johnson, and Bill Tomlinson. "Integrated Learning for Interactive Synthetic Characters." *ACM Transactions on Graphics* 21, no. 3: Proceedings of ACM SIGGRAPH 2002 (2002).

[5] Bolhuis, Johan J., and Jerry A. Hogan. *The Development of Animal Behavior: A Reader.* Oxford: Blackwell Publishers, 1999.

[6] Coppinger, Raymond, and Lorna Coppinger. *Dogs: A Startling New Understanding of Canine Origin, Behavior, and Evolution.* New York, NY: Scribner, 2001.

[7] Dayan, Peter, and Geoffrey E. Hinton. "Feudal Reinforcement Learning." In *Advances in Neural Information Processing Systems 5.* San Mateo, CA: Morgan Kaufmann, 1993.

[8] Dietterich, Thomas G. "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition." *Journal of Artificial Intelligence Research* 13 (2000): 227-303.

[9] Fentress, John C., and Peter J. McLeod. "Motor Patterns in Development (except)." In *The Development of Animal Behavior: A Reader*, edited by Johan J. Bolhuis and Jerry A. Hogan. Oxford: Blackwell Publishers, 1999.

[10] Hall, William G., and Christina L. Williams. "Suckling Isn't Feeding, or Is It?" In *The Development of Animal Behavior: A Reader*, edited by Johan J. Bolhuis and Jerry A. Hogan. Oxford: Blackwell Publishers, 1999.

[11] Hogan, Jerry A. "Development of Behavior Systems." In *Handbook of Behavioral Neurobiology*, edited by E.M. Blass, 229-79. New York: Kluwer Academic Publishers, 2001.

[12] Humphrys, Mark. "Action Selection Methods Using Reinforcement Learning." In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, edited by P. Maes, M. Mataric, J.-A. Meyer, J. Pollack and S. W. Wilson, 135-44. Cambridge, MA: MIT Press, Bradford Books, 1996.

[13] Kaelbling, Leslie Pack. "Hierarchical Learning in Stochastic Domains: Preliminary Results." In *Proceedings of the Tenth International Conference on Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.

[14] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement Learning: A Survey." *Journal of Artificial Intelligence Research* 4 (1996): 237-85.

[15] Kruijt, Jaap P. "Ontogeny of Social Behavior in Burmese Red Junglefowl (*Gallus Gallus Spadiceus*) (except)." In *The Development of Animal Behavior: A Reader*, edited by Johan J. Bolhuis and Jerry A. Hogan. Oxford: Blackwell Publishers, 1999.

[16] Lin, Long-Ji. "Reinforcement Learning for Robots Using Neural Networks." PhD thesis, Carnegie Mellon University, 1993.

[17] Lorenz, K., and P. Leyhausen. *Motivation of Human and Animal Behavior: An Ethological View*. New York, NY: Van Nostrand Reinhold Co., 1973.

[18] McCallum, Andrew K. "Reinforcement Learning with Selective Perception and Hidden State." PhD thesis, University of Rochester, 1996.

[19] Parr, Ronald, and Stuart Russell. "Reinforcement Learning with Hierarchies of Machines." In *Proceedings of Advances in Neural Information Processing Systems 10*. Cambridge, MA: MIT Press, 1997.

[20] Singh, Satinder P. "Learning to Solve Markovian Decision Processes." PhD thesis, University of Massachusetts, 1994.

[21] Sun, Ron, and Todd Peterson. "Multi-Agent Reinforcement Learning: Weighting and Partitioning." *Neural Networks* 12, no. 4-5 (1999): 727-53.

[22] Sun, Ron, and Chad Sessions. "Automatic Segmentation of Sequences through Hierarchical Reinforcement Learning." In *Sequence Learning: Paradigms, Algorithms, and Applications*, edited by Ron Sun and C. Lee Giles. Berlin; Heidelberg: Springer-Verlag, 2000.

[23] Tham, Chen K., and Richard W. Prager. "A Modular Q-Learning Architecture for Manipulator Task Decomposition." In *Proceedings of the Eleventh International Conference on Machine Learning*. New Brunswick, New Jersey: Morgan Kaufmann, 1994.

5731
65