The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots

Harold Henry Chaput

Report TR04-34 August 2004

hhchaput@mac.com
http://homepage.mac.com/hhchaput/

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

The Constructivist Learning Architecture:
A Model of Cognitive Development for Robust Autonomous Robots

by

Harold Henry Chaput, B.A., M.S.C.S.

Dissertation

Presented to the Faculty of the Graduate School of

the University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2004

This dissertation is dedicated to my parents, Henry and Constance Chaput.

Acknowledgements

This dissertation would not have been possible without the guidance, patience and brilliance of my amazing dissertation committee. An eternal debt of gratitude goes to Ben Kuipers, whose inspirational words were the genesis of this work; Les Cohen, whose sage and keen insight showed me the value of my own ideas; and Risto Miikkulainen, who showed me the path from cool ideas to compelling science. I also want to thank Bruce Porter and Ray Mooney for their advice and encouragement, and Peter Stone for some great ideas and Stéphane Grappelli. I am ever thankful.

It was really Robert Turknett who goaded me into working on this topic, and many of the ideas in this dissertation came from our lofty conversations and heady debates. Rob is my foil, my straight man, and my big brother. Thanks pal.

I would have been completely lost in Psychology were it not for Cara Cashon, who was exceptionally nice and patient with me from the moment we met. She has a radiant intellect coupled with a vast heart. I am grateful to count her among my close friends.

The UTCS department is full of cool people with big brains, but Ken Stanley, Jeff Provost, and Tal Tversky were my brothers in arms and partners in crime. Each one is a genius and an exceptional human being. Somebody give them a job.

I am blessed with the greatest friends a man could have. I couldn't have written this dissertation without Adam Biechlin, Sue Blankenship, Carmen Gonzalez-Sifuentes, Cinqué Hicks, Mark Holzbach, Andy Hundt, Bob Kaeding, Cheyenne Kohnlein, Charles Lewis, Manuel and Davíd Quinto-Pozos, Rich Skrenta and Joel Stearns. And a special expression of love to Jennifer Potter, the sister I never had. Drinks all around.

I am deeply grateful to my comrades at the Austin Museum of Digital Art, especially Todd Simmons, Ginny Blocher, Kat Sauceda and Lisa Lee. AMODA was more than just a distraction from my research, it was nourishment for the other half of my brain. This research might be more complete if it wasn't for AMODA, but it would also be duller. Every AMODA volunteer is an inspiration, and I am honored to have worked with them. Thanks for playing museum with me.

Dr. Brian Slator was the man who made me serious about going to grad school in the first place, and he has treated me like a son ever since. He, along with Rita, Audrey and Megan, cared for me and made it impossible to imagine a future without my Ph.D. I couldn't have lucked into a better councillor or role model, and I hope I grow up to be just like him. Rock on.

All these people got me started and helped me along the way. But nobody is more responsible for motivating me to finish than Kazuki Kinjo, the rarest of gems. Our's is a most impractical, improbable, thoroughly modern romance, one that has inspired this author to bridge oceans, challenge states, and move mountains (including the one in your hands). Only Kazuki could convince me that there is life — and love — after the defense, and I can't wait to get there. 愛する人よ、ありがとう。

The Constructivist Learning Architecture: A Model of Cognitive Development for Robust Autonomous Robots

Publication No. _____

Harold Henry Chaput, Ph.D.
The University of Texas at Austin, 2004

Supervisors: Benjamin J. Kuipers and Risto Miikkulainen

Autonomous robots are used more and more in remote and inaccessible places where they cannot be easily repaired if damaged or improperly programmed. A system is needed that allows these robots to repair themselves by recovering gracefully from damage and adapting to unforeseen changes. Newborn infants employ such a system to adapt to a new and dynamic world by building a hierarchical representation of their environment. This model allows them to respond robustly to changes by falling back to an earlier stage of knowledge, rather than failing completely. A computational model that replicates these phenomena in infants would afford a mobile robot the same adaptability and robustness that infants have. This dissertation presents such a model, the Constructivist Learning Architecture (CLA), that builds a hierarchical knowledge base using a set of interconnected self-organizing learning modules. The dissertation then demonstrates that CLA (1) replicates current studies in infant cognitive development, (2) builds sensorimotor schemas for robot control, (3) learns a goal-directed task from delayed rewards, and (4) can fall back and recover gracefully from damage. CLA is a new approach to robot control that allows robots to recover from damage or adapt to unforeseen changes in the environment. CLA is also a new approach to cognitive modeling that can be used to better understand how people learn for their environment in infancy and adulthood.

Contents

List of Tables

List of Figures

1. Introduction

Figure 1.1: Mars rover Spirit. The autonomous robot sent to explore Mars. Robots are being used more for autonomous operation far from their human operators, so they need to be resilient to unforeseen problems. A learning system to address this need is the subject of this dissertation.

Robots are being employed for tasks that allow minimal human interaction and accessibility (Figure 1.1). As these robots become more autonomous, it is more important for them to be more flexible in the face of unplanned conditions, adaptive to unforeseen changes, and robust during unexpected problems. For a robot to meet these requirements, it must have a controller that can adapt to its environment and retrain itself when the environment changes.

This dissertation solves this problem by looking to infant cognitive development. Young infants learn a great deal from their environment that they use throughout their lives. They are constantly adapting to new features in the world starting with the simplest perceptions of the world, and respond to new and unexpected information through adaptation.

This introduction motivates the work presented later by describing the problem of robust autonomy, and how it this problem is similar to the issues faced by young infants. The approach to this problem is then proposed: by building a learning system that models infant cognitive development, this system can be used by an autonomous robot for robust control.

1.1 Motivation

As robots become more sophisticated, articulated and affordable, they are being used with increasing frequency for tasks that would be too costly or hazardous for a humans to perform. Robots today can carry an impressive array of tools and perform tricky physical tasks. Remote controlled robots are used to disarm bombs, keeping human technicians at a safe distance. Sub-

mersible robots are sent to explore the ocean floor, which is inhospitable to human explorers. Robots are sent into buildings damaged by earthquakes to locate people who are trapped, and free them. Rather than incur the expense and potential loss of life that human space travel might bring about, robots are sent to Mars to explore and report findings back to scientists on Earth. Robots are getting better, cheaper, and are expendable, making them a good alternative to human explorers and technicians.

Often, these applications require that robots not only perform sophisticated tasks, but do so with limited human intervention, and with limited foreknowledge of the robot's eventual operating environment. For example, when NASA designed Spirit and Opportunity (the robots sent to explore Mars; Figure 1.1) they had to take into account the 10 minute signal transmission time from Earth to Mars, and another 10 minutes to send a signal back. This time delay makes direct control of the rovers untenable: a robot that is rolling along towards a drop-off would not receive a command to stop or turn until 20 minutes later, at which time it could be in pieces at the bottom of a cliff. Consequently, these robots operate autonomously by sensing their environment, seeking goals, and avoiding dangers with minimal human intervention.

1.1.1 Importance of Flexibility

Perhaps the biggest issue with autonomous robotics is flexibility. In addition to performing its task and avoiding dangers, an autonomous robot must handle situations that its engineers did not foresee. For example, the Martian terrain could be softer than had previously been known, which would change the way the robot moves around. The robot could wander into the shade of an outcropping of rock, making objects look different to the cameras than they do in direct sunlight. Or the robot could sustain some damage during its operation, knocking the camera to the left of center and shifting the robot's view of the world. All of these issues would impact the performance of the robot, and the robot's engineers must predict every possible problem and design its controller to operate under these new circumstances. How flexible the robot's controller is determines how well the robot would continue operating. If the change is unexpected and significant, the robot could easily be rendered inoperative or unusable for the mission. Since the reason the robot was sent to Mars in the first place was because it was too difficult to send people, there is little chance that a technician could be sent out for some field repair.

Consequently, much of the engineering of autonomous robots involves preparing for the unexpected. Engineers must conceive every possible problem the robot might encounter, and develop a controller that can account for all of them. This approach has two drawbacks. First, a controller that has specialized code for every possible problem will, under normal circumstances, be largely unused. Even if something goes wrong, only the part of the controller that handles the specific problem will be utilized, while the rest goes unexecuted. Not only is this an issue of computational resources, but of the amount of time spent by engineers trying to see into the future when they could be making the robot better in other ways. Of course, some amount of exception handling is always included in even the simplest software, but there is clearly a trade-off between robustness and resources. Second, it is not possible to plan for unforeseen problems. Robots are complex systems and many robot environments, like Mars, are complex as well. It is unlikely that a team of engineers will account for every possible problem the robot might encounter. No matter how much planning goes into building the robot controller, there will always be circumstances that the robot engineers did not plan for. And the problem for which there is no plan then renders the robot inoperative.

One solution to this problem is to let the robot figure out its own solution to any problem it might encounter. The robot already has the autonomy to pursue its goals and avoid dangers, so it is a natural extension to allow the robot to adapt to its environment. A robot that adapted to changes in the environment would not only discover solutions to problems, but would discover better, unforeseen ways of achieving its goals. Instead of building a controller to handle every possible problem, engineers can focus on the single problem of how the robot can adapt and let the robot re-program itself by learning its own model of the world, and by building its own controller to achieve goals and avoid danger.

How does one create a learning system that develops its own model of the world, but is also sophisticated enough to handle complex tasks efficiently? Fortunately, this is a problem that has already been solved; indeed, it has already been solved by each of us. Every human infant faces this same issue at birth: they are introduced to a world that is entirely new to them, and must build up a world model that allows them to perform complex tasks. Infants are constantly introduced to new information in the world, and all the while their bodies are changing and growing. Infants are famously adept at solving problems and overcoming obstacles. The learning system used by infants addresses all of the requirements of an adaptable, autonomous learning system. Therefore, a learning system that models infant cognitive development will also give a robot robust autonomy. This conjecture is the central hypothesis of the dissertation.

1.1.2 Fallback and Recovery

One particularly useful technique used by infants to achieve adaptive, robust behavior is fallback. An infant has a partially learned a skill set, but is confronted with a situation which its nascent skills cannot handle. Rather than fail completely, the infant will utilize an earlier skill set (Cohen and Oakes 1993). Eventually, infants will integrate the novel situation into its skill set and recover its ability to cope with it. Fallback is an important feature for a robot controller because the robot's performance can gracefully degrade under suboptimal conditions, rather than completely fail.

To illustrate the concept of fallback, consider the following example. (This is not a scientific study, simply an illustration of the fallback principle at work.) Amateur typists start out using the "hunt and peck" method: they search for a the first letter, press the key, then search for the next letter, press the key, etc. As typists become more experienced, they become more familiar with the finger movements needed frequent letter combinations and eventually whole words and phrases. Typing a pervasive word like "the" becomes one swift action when the fingers move almost in parallel on the keyboard.

When presented with a keyboard with different dimensions, such as a thumbboard, the experienced typist is presented with a challenge: all of the learned procedures to move fingers certain distances, and the sensory feedback from the fingers that guide the typing process, are useless because the thumbboard cannot accommodate all ten fingers like a full-sized keyboard can. But part of what makes thumbboards successful is that they are actually easy to use for experienced typists: all of the skill is not lost. The skill of typing does not depend entirely on the direct motor commands to the fingers and the exact feedback from the hand, but has many layers from the concrete — such as muscle contractions in the finger — to the abstract, like the combination of finger movements needed to type "the." Therefore the typist can *fall back* to a lower, less abstract level of skill to achieve the goal. Of course, there is an initial degradation in performance, but eventually

the typist will *recover* the full typing skill as lower-level knowledge develops new high-level skills specific to the thumbboard.

Now consider a more radical change to the keyboard configuration. Imagine that the typist must now move from a standard "QWERTY" keyboard to a modern "Dvorak" key configuration. Now the typist is faced with a more fundamental challenge, and cannot even rely on knowledge of relative key positions. The typist must fall back to an even lower level of skill, perhaps all the way back to the "hunt and peck" approach. Even so, the typist will still recover as new knowledge is built on these lower-level skills and, eventually, the typist will become expert on this new keyboard configuration.

Fallback is a strategy for dealing with a change in the environment, small or large, by allowing agents to respond by falling back to a lower level of skill, rather than completely failing. Fallback is usually accompanied by a degradation in performance, but only partial or "graceful" degradation. Eventually, the agent will recover from the skill degradation and once again become an expert at the task. This is a technique that infants use to cope with a new and ever changing environment. It is also the technique that is used to give robots robust control.

1.2 Approach

To build a computational model of infant cognitive development, the model must be based on evidence of the learning mechanism used by infants. Piaget (1936; 1937) proposed a theory for infant cognitive development called constructivism, which describes the part-to-whole system used by infants to integrate simple ideas into increasingly more complex ones. Constructivism is a powerful theory of infant cognition, but the details of the learning mechanisms in constructivism have long remained a mystery. Since Piaget, new techniques have been developed that have brought about a watershed of new empirical evidence of infant cognition and cognitive development. This evidence has been integrated to form a set of Information Processing Principles (Cohen, Chaput & Cashon 2002) which outline, at a high level, the nature of the learning mechanism used by infants.

This new empirical evidence allows for a model of infant cognitive development to be built. This model, the Constructivist Learning Architecture (CLA) is first presented in this dissertation. CLA is an unsupervised learning system that builds a hierarchical knowledge base from observing the environment. CLA is built by interconnecting a hierarchy of Self-Organizing Maps (SOMs; Kohonen 1997). SOMs are chosen because they are unsupervised, self-organizing learning systems that have previously modeled both cognition and neuroscience.

CLA is shown to be a model of infant cognitive development by reproducing studies in the challenging domain of causal acquisition. While the prevailing view had been that infants can innately perceive causal events (Leslie 1984), recent studies suggest that causal perception is learned (Cohen & Amsel 1998). The purpose of this experiment is not just to show that CLA can learn what infants learn. More important is that CLA learn in the same way as infants — by building high-level concepts from lower-level knowledge — and fail in same way as infants — by falling back to a prior skill set.

To operate as a robot controller, CLA must move beyond self-organization of perceptual stimuli and build sensorimotor schemas that associate perceptions with actions. Such a system was proposed by Drescher (1991), and his Schema Mechanism had stood as the only constructiv-

ist sensorimotor schema building system. This dissertation presents a technique for using CLA to build a hierarchy of sensorimotor schemas that is more efficient the Schema Mechanism and, more importantly, is the first such implementation that also models infant cognition.

Finally, to achieve goal-directed behavior, a technique is introduced whereby CLA can learn from delayed rewards. This technique allows a robot to use CLA as a controller by building a hierarchy of sensorimotor schemas, and linking the schemas together to build a plan for achieving a goal. CLA represents a complementary approach to Reinforcement Learning (Sutton & Barto 1998) that elaborates the agent's state-space by constructing new features. Ultimately, CLA makes a robot robust by utilizing CLA's fallback ability. Building a model of infant cognitive development and using it as a robot controller will give a robot the grounded knowledge, adaptation and robustness of an infant.

1.3 Overview of the Dissertation

This dissertation consists of four parts: Introduction and Background (Chapters 1 and 2), CLA (Chapter 4), Applications (Chapters 5, 6 and 7), and Evaluation (Chapters 8 and 9). Because this work spans both computer science and psychology, related work is discussed in each chapter, where appropriate.

Chapter 2 reviews research in psychology and computer science upon which the work in this dissertation is based, including Piaget's (1936; 1937) theory of constructivism and the Information Processing Principles (Cohen, Chaput and Cashon 2002). Also discussed is the Self Organizing Map (SOM; Kohonen 1997), a central technology used in the Constructivist Learning Architecture.

Chapter 3 describes the Constructivist Learning Architecture, a model of infant cognitive development used by robots for robust autonomy. This is the main contribution of this dissertation.

In **Chapter 4**, CLA is demonstrated as a model of infant cognition by modeling an infant's acquisition of causal perception. CLA is shown to be robust when presented with noisy data. CLA is also compared to other models of infant cognition.

In **Chapter 5**, CLA is shown to be a robot controller. CLA reimplements the Schema Mechanism (Drescher 1991), a constructivist controller for autonomous robots. CLA replicates all the functionality of the Schema Mechanism, but more efficiently. Moreover, CLA is shown not only to be a cognitive model, but also a learning system that can control a mobile robot.

Chapter 6 combines the ideas of the two previous chapters and applies CLA to a realistic robot and environment. CLA is used by a robot to learn how to forage for specimens. CLA learns hidden features in the environment, and recovers gracefully when artificially damaged. CLA is also compared to other robot controllers and systems that learn using delayed rewards.

Chapter 7 discusses the accomplishments of this dissertation and suggests some potential future work that can come from it. Finally, **Chapter 8** presents an overview of the work presented in this dissertation and concludes.

2. Foundations

This dissertation describes a computational model of infant cognitive development which is also used by mobile robots for robust autonomous control. Two areas of research — infant cognition and machine learning — form the basis of the model. This chapter provides an overview of the foundations in these two areas.

The central thesis of the dissertation is that a model of infant cognitive development can be used to establish robust control for autonomous robots . To provide a background for modeling infant cognitive development, this chapter first discusses the subject of infant cognition, starting with Piaget's (1952) theory of Constructivism, and moving through a body of contemporary studies in infant cognition and cognitive development. The section concludes with a look at a new approach to infant cognitive development, the Information Processing Principles, which forms the basis of the model presented in this dissertation.

The model presented here is based on previous work in machine learning. Specifically, the model uses the Self-Organizing Map (SOM; Kohonen 1997) as a central computational unit. The second section will describe the SOM, detail how the SOM learns, and discuss its relevance to the area of infant cognitive development.

In the following chapter, these two scientific areas are merged to form the computational model that is the main contribution of this dissertation: the Constructivist Learning Architecture, or CLA.

2.1 Psychological Foundations

The theoretical foundation of this dissertation comes from the field of infant psychology. This section provides an overview of some important research in this field, starting with Piaget's (1936; 1937) theory of Constructivism, moving to more modern studies in infant cognition, and finally to an Information Processing approach to infant cognitive development. This section includes a description of the Information Processing Principles (Cohen, Chaput & Cashon 2002) which are the central ideas upon which this research is based.

2.1.1 Constructivism

The modern study of infant cognition can truly be said to have started with the research of Piaget (1936; 1937). His seminal work in the area of infant cognition presented for the first time a

comprehensive picture of infant cognitive development, and provided a theory of infant cognition called Constructivism.

Constructivism describes, among other things, a stage-like developmental pattern that Piaget observed in infants. Piaget theorized that infants had knowledge in the form of schemas through which the infant could process and comprehend its environment. Schemas could be propositional schemas that relate and classify observations, sensorimotor schemas that relate observations to actions, or operational schemas that describe systems and their inner workings. Operational schemas come later in an infancy's development, but propositional and sensorimotor schemas are constructed immediately and play a central role in cognitive development. (There will be examples of both propositional and sensorimotor schemas in later chapters.)

The developmental aspect of schemas has two parts: assimilation and accommodation. In assimilation, the infant attempts to apply what it knows (that is, its schemas) to everything in its world. All observations are viewed and processed in terms of the infant's schemas. This process is characterized as schemas assimilating the world.

However, the world cannot always be addressed by the infant's complement of schemas. There are limits to the applicability of any given schema, and some things in the infant's environment may not apply to any of the infant's schemas. Piaget characterized this as "resistance to the external world." In response, the infant must accommodate the world, which is accomplished by creating a new set of schemas that describe those aspects of the world that are not addressed by the existing set of schemas.

Since schemas assimilate every object, they also assimilate other schemas in addition to direct environmental observation. Thus, when new schemas are built to accommodate the world, they take into account not only the infant's experience with the environment, but preexisting schemas as well. In other words, new schemas are constructed using old schemas. It is this part-to-whole progression — the essence of constructivism — that allows infants to start with a very simple view of the world and build a complex, hierarchical knowledge base for mature behavior.

2.1.2 Contemporary Studies in Infant Cognition

While Piaget (1936; 1937) provided a comprehensive theory of infant cognitive development, the field lacked the necessary techniques for probing the details of cognitive development in infants, and he was unable to provide a formal model of constructivism. Consequently, the field of infant cognition eventually responded with a more nativist view.

Not until the advent of the habituation technique were researchers able to explore the details of infant cognition. The habituation technique relies on a novelty preference in infants. When an infant is presented with the same familiar scene again and again, the infant will grow bored and look away from the scene, presumably in search of something new. If, however, the infant is presented with something new, the infant will look longer. Thus, looking time can be used as a measure of novelty.

An experimenter, then, can design a habituation experiment using a scene that can be familiar in one way, and novel in another way. Such an experiment can be used to determine how the infant is processing the scene.

For example, Cohen and Younger (1984) used the habituation technique to determine how infants processed the visual perception of angles. The stimuli they used is reproduced in Figure 2.1. In the study, infants were repeatedly presented with a 45° angle Training Stimulus until the infant had habituated and started looking less and less. The infant was then presented with one of

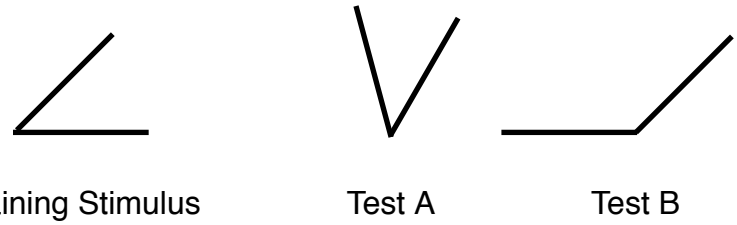

Figure 2.1: Stimuli for an infant habituation experiment. These stimuli were used by Cohen and Younger (1984) to determine how infants cognitively process angles. The Training Stimulus is a 45° angle, which the infants would be see until they habituated. Then infants were presented with one of two test stimuli. Test A is also a 45° angle, but the line segments that make up the angle are at a different orientation from the Training Stimulus. Test B retains the orientation of the line segments in the Training Stimulus, but they are configured to make a 135° angle. Infants that process angles as a whole will see Test A as similar to the Training Stimulus, and thus dishabituate more to the novel angle in Test B. Infants that process the stimuli as two individual line segments will recognize the segments in Test B and respond more to the new orientation of Test A. The habituation experiment is a useful tool for probing the nature of an infant's perception and cognition.

two different test stimuli. One test stimulus was another 45° angle, but one that had been rotated so that the line segments that made up the angle were at a different orientation than the Training Stimulus. The other test preserved the orientation of the line segments, but connected them to create a 135° angle. If the infant processed the stimuli using primarily the angle information, then the infant will recognize the 45° angle as similar to the Training Stimulus, and not respond very much, while at the same time the 135° angle will appear novel — despite having the same line segment orientations — and will cause the infant to dishabituate. On the other hand, if the infant processed the objects as individual line segments, then the rotated 45° angle would appear to be completely new, while the 135° angle would be familiar.

New Developmental Evidence

With the introduction of the habituation technique, most contemporary studies of infant cognition focussed on providing experimental proof of the current nativist theories. Many studies postulated the that much of infant knowledge was innate or precocious, such as the ability to process object unity (Kellman and Spelke 1983), object persistence (Baillargeon 1987; Baillargeon, Spelke & Wasserman 1985), object individuation (Wilcox and Baillergeon 1998), and object solidity (Spelke et al 1992).

Many of these studies, though, relied on testing infants only at a single age, and often overinterpreted the results. In response, other researchers performed follow-up studies on infants at multiple ages and found evidence that infants exhibited a developmental progression in these and many other areas (see Cohen & Cashon 2003 for an overview). For example, Cohen and Younger (1984), using the stimuli mentioned above, determined whether infants at different ages perceived angular relations. They found that infants at 14 weeks would process the stimuli as whole angles, ignoring the orientation of the line segments. However, 6-week-olds would respond to the line segment orientation, and be unresponsive to the angle information. While these findings are not definitive proof that infants learn this knowledge, it does make the nativist explanation less parsimonious. This same technique was applied to find developmental trends in many domains, including object unity (Slator, Morison et al 1990; Johnson & Nañez 1995), object persistence (Bogartz, Shinskey & Schilling, 2000; Cashon & Cohen, 2000; Schilling, 2000), object individuation (Need-

ham 2001), object solidity (Cohen, Gilbert and Brown 1996), facial perception (Cohen & Cashon 2001), and causality (Cohen, Amsel, Redford, and Casasola 1998).

2.1.3 An Information Processing Approach to Cognitive Development

With the discovery of developmental trends in so many different areas, a pattern started to emerge among all the studies. Regardless of the domain, the method of development appeared to be similar in each case, and it resembled the constructivist theory of Piaget. However, unlike before, these studies uncovered more details of the developmental process, and a sharper picture of infant cognitive development could be drawn. From these studies came an Information Processing approach to infant cognitive development (Cohen 1998), which postulated that much of infant knowledge thought to be innate can actually be accounted for by processing information in the environment. Additionally, much of the observed developmental phenomena — such as stage-like development, or the so-called U-shaped curve — can be explained by the self-organization methods used to process information, rather than through external forces, such as physiological development or normative feedback.

The Information Processing approach to infant cognitive development is essentially a re-statement of Piaget's constructivism framed in the context of modern learning systems. This approach can be summed up using six Information Processing Principles (Cohen, Chaput & Cashon 2002), which are listed below. These principles were arrived by examining developmental changes in several different topics considered to be aspects either of infant perception or infant cognition. These are:

1. Infants are endowed with an innate information-processing system.
Infants are born neither with a blank slate nor a preponderance of innate core knowledge. Rather, infants are born with a system that enables them to learn about their environment and develop a repertoire of knowledge. From the outset, the innate system provides architectural constraints in how this learning may be accomplished. The system is designed to allow the young infant access to low-level information, such as orientation, sound, color, texture, and movement.

2. Infants form higher schemas from lower schemas.
In other words, the learning system is hierarchical. As the infant learns and develops, information that is accessed becomes more and more complex, building upon prior processed information. An assumption underlying this principle is that the ability to process more complex information is the result of being able to integrate the lower-level schemas into a more complex, higher-level schema. That integration is based upon statistical regularities or correlations in activity of those lower-level schemas.

So, to refer back to the angle study (Cohen & Younger 1984), an infant may initially process the two lines of a 45° angle as separate units in particular orientations, but because the two lines co-occur in the same relative spatial relationship, even when the angle is rotated, the infant will eventually process the relationship among the lines, that is, the angle rather than the independent lines.

3. Higher schemas serve as components for still-higher schemas.
The hierarchical nature of the system can account for development beyond the first few months of life. The process of integrating information to form a higher-level schema is itself repeated throughout development. Lower-level information can be integrated into a higher-level schema, which can in turn be integrated into an even higher-level schema, and so on.

To continue with the angle example, after connections have been formed between the two lines to form an angle, several angles and curves could then be integrated to form the complex shape of an object. That object could then be integrated with another object to form an event defined in terms of the relationship between the two objects.

4. There is a bias to process using highest-formed schemas.
Whereas the previous principles have described the learning mechanism, or the building of the hierarchy, this fourth principle describes what information an infant will attempt to process after two or more layers of information units have been formed. Specifically, infants will tend to process the incoming information using the highest level available to them. Preferring a higher level does not mean that the lower-level information is unavailable, but rather that the most adaptive strategy for an infant is usually to process information at the highest possible level. The next principle describes what happens when that strategy fails.

5. If, for some reason, higher schemas are not available, lower-level schemas are utilized.
This principle describes the "fallback" described in section 1.1.2. A higher-level schema may be unavailable for a number of reasons, but it is often unavailable when the system gets overloaded. Circumstances for overloading the system can vary, but may include complicating the input by adding irrelevant material or noise, or converting a simple object or event into a category of objects or events. A corollary of this principle is that if for some reason the system does fall back to a lower level, it will then attempt to learn or accommodate so as to move back up to the next higher level.

6. This learning system applies throughout development and across domains.
A strength of any theory is that it can account for a variety of findings in a variety of domains. This final principle highlights the domain-general nature of the proposed learning system. Although these principles were developed to explain how infants develop cognitively, they are likely a great deal more general. They may, in fact, represent how we as humans become proficient or experts in a wide range of tasks throughout the life span. It just so happens that one of first tasks encountered by young infants is trying to make sense of the immediate physical and social world around them. These principles help them succeed at that task.

2.1.4 Conclusion

The Information Processing Principles (Cohen, Chaput & Cashon 2002) describes a domain-general learning system that provides continuous learning from the environment, the application of learned schemas to achieve some goal, and a fallback method for graceful recovery in the event of information overload. These are all features that are useful — perhaps vital — to robust autonomous robot control. A computational learning system that can capture these principles can be used by a robot for grounded and robust control. The goals of this dissertation are to build such

a model, and apply it to a mobile robot for grounded, robust control. This model is presented in the next chapter.

2.2 Computer Science Foundations

The computational model presented in the next chapter uses a learning system called the Self-Organizing Map (SOM; Kohonen 1997) as a central module. Before a description of the new model can commence, the SOM is first described in this section. While other learning systems might work just as well as the SOM for this dissertation, the SOM is an appropriate tool for a variety reasons. These reasons, along with a technical description of the SOM, follow.

2.2.1 The Self-Organizing Map (SOM)

The Self-Organizing Map (SOM; Kohonen 1997) is an unsupervised learning system that maps input data onto a feature coordinate system, or feature map. The input data is a collection of feature vectors that describe observations of the environment. The SOM itself consists of a map of interconnected nodes, where each node contains a reference vector $m_i = [\mu_{i1},\mu_{i2},...,\mu_{in}]^T \in \Re^n$. The input vector $x = [\xi_1,\xi_2,...,\xi_n]^T \in \Re^n$ is connected to all the nodes i in parallel with through the weight vector μ_{ij}.

For training, the SOM first finds a best matching node by first looking for the node whose reference vector is most similar to the input vector. To do this, the SOM utilizes a distance metric d to determine the similarity of an input vector x with a given node m_i. Usually, a standard Euclidean distance $\|x - m_i\|$ is used, but there are a variety of distance metrics that could be used for their particular properties.

The distance metric is used to find the closest node m_c, sometimes called the "winning node." Locating the winning node is done by finding c such that:

$$c = \arg\min_i \left\| x - m_i \right\|. \qquad 2.2$$

During learning, the winning node is selected and becomes the focal point for the modification of the reference vectors. The nodes in the SOM are made to approach the input vector proportional to the node's topological distance to the winning node in the SOM:

$$m_i(t+1) = m_i(t) + \alpha(t) \cdot h_{ci}(t)\left[x(t) - m_i(t) \right], \qquad 2.3$$

where t is an integer discrete-time coordinate, α is the learning rate, and h_{ci} is a neighborhood function. h_{ci} determines how the rate of change falls off as the distance to the winning node increases. A simple neighborhood function would define a set of nodes around the winning node that would receive training, while nodes outside this set would be unchanged. Another method, used in this dissertation, defines the neighborhood function using a Gaussian, where $\sigma(t)$ represents the size of the neighborhood:

$$h_{ci} = \exp\left(-\frac{\|r_c - r_i\|^2}{2\sigma^2} \right). \qquad 2.4$$

To facilitate convergence of the SOM, $\alpha \to 0$ and $h_{ci} \to 0$ as $t \to \infty$.

2.2.2 Features of the SOM

As the SOM is trained on input vectors, the reference vectors come to represent clusters of the feature vectors that exist in the environment. In other words, the reference vectors become prototypes for the stimuli of the input vectors. The SOM is a learning system for generating prototypes from the environment without the use of a corrective error signal — in other words, it is unsupervised.

The SOM is also sensitive to frequency in the input set. Similar stimuli that are presented more frequently will get more representation in the SOM, which a greater number of nodes to that stimulus cluster. Each prototype represents a subset of the input vectors proportional to its frequency in the environment, so more frequent stimuli will be classified with greater accuracy. The SOM effectively magnifies dense clusters to provide differentiation where it is most needed.

The neighborhood effect of the SOM results in similar prototypes being topologically proximal. Insofar as the distance metric measures the similarity of the stimuli, clusters of nodes will represent a family of similar features in the environment.

Finally, the SOM is rooted in a body of supporting psychological and neurological evidence. There have been many studies comparing the SOM to cortical maps of neurons. And the SOM has been used to successfully model many psychological phenomena, including vision, audition, voicing and language (for an overview of this evidence, see Kohonen 1997).

2.2.3 The SOM for Cognitive Development

The SOM's ability to generate prototypes unsupervised makes it highly suitable for modeling cognition. First, the SOM provides a reasonable and psychologically supported way of finding correlations among features in the environment, making the SOM a useful tool for building schemas.

The Information Processing Principles appear to describe a self-organizing learning system. Nothing in the principles rules out the use of a supervised learning system, but neither do they mention any attributes specific to supervised learning systems, such as normative feedback or error correction. Also, some systems using normative feedback have been criticized when applied to some domains of infant cognitive development (this is discussed in more detail in section 4.5.1). While supervised learning systems may work well as a model of cognitive development, the SOM does not rest on this assumption. Additionally, using the SOM avoids the criticism (fair or not) that a supervised learning system would receive.

The SOM's neural plausibility also makes it a good candidate for a model of infant cognition. While it is not necessary for the model to be neurally plausible in order to be psychologically valid, it does address the concerns of many critics who are wary of groundless psychological models. The SOM's neural plausibility makes it — and anything built with it — more attractive and influential in infant cognition, and psychology in general, than a learning system that has no relation to neuroscience.

2.3 Conclusion

This chapter presented the two scientific foundations of the research in this dissertation. First, a body of research on infant cognition suggests an information processing mechanism for cognitive development. This mechanism builds a hierarchy of knowledge by observing the environment, it is domain general, and it provides a fallback mechanism for graceful recovery in the event of overload. Second, the SOM provides an unsupervised learning system for building prototypes through observations of the environment. The SOM is an unsupervised, neurally plausible correlation learning system, making it a desirable system to use in a cognitive model.

A computational model of infant cognitive development — specifically the Information Processing Principles — can be used to give a robot grounded, robust autonomous control. The SOM goes a long way towards implementing the Information Processing Principles, but is missing certain key features. In the next chapter, these features are addressed and a computational model of the Information Processing Principles using the SOM is presented.

3. The Constructivist Learning Architecture

The Constructivist Learning Architecture (CLA, pronounced "clay" as in "modeling clay") is a learning system that models infant cognitive development and provides adaptive, robust control for autonomous robots. To make CLA a model of infant cognitive development, CLA implements the Information Processing Principles presented in Chapter 2, which were derived from an extensive analysis of infant cognition studies. By basing CLA on the Information Processing Principles, CLA can replicate studies of infant cognition, as will be demonstrated in Chapter 4. Further chapters will show that such a learning system can be used to control an autonomous agent (Chapter 5) and develop robust control while learning from delayed rewards (Chapter 6).

This chapter contains a detailed description of the design and implementation of CLA, the main contribution of this dissertation. The first section addresses the design decisions that were made for building CLA. The second section describes the CLA implementation, including activation and training algorithms. The final section explores CLA's ability to model different information architectures.

3.1 The CLA Design

The goal of CLA is to implement a model of constructivist learning in infants that can be utilized as a robot controller. The motivation behind this goal is to create a system that captures the adaptability and robustness of infants and apply these features to mobile robot control. To make CLA a model of constructivist learning, CLA is implemented using the Information Processing Principles (Cohen, Chaput & Cashon 2002) as a design specification.

Such a system is necessarily hierarchical. Principles 2 and 3 describe a system where schemas are processed and become new schemas, which suggests not only a hierarchy, but one where the inputs and the outputs have the same representation type. Principles 4 and 5 describe the ability to utilize different levels of the representation as needed, meaning that the implementation cannot throw away the lower schemas once the newer schemas are created. So the implementation must be a persistent hierarchy of interconnected modules.

The Self Organizing Map (SOM) (Kohonen 1997) was chosen as the basis for the implementation of CLA because it is an unsupervised, neurologically plausible system for learning prototypes, as discussed in the previous chapter. But the SOM by itself is not enough to implement the Information Processing Principles. In particular, the SOM is a flat learning system that projects the

feature space of the stimuli onto a two-dimensional space. Additional work must be done to give the SOM the ability to build a hierarchical knowledge base.

3.2 CLA Implementation

The challenge of making the SOM hierarchical lies in the problem of how to generate output from a SOM that can be used as meaningful input to another SOM. The approach used here is to use the nodes of a trained SOM as new feature detectors, similar to those used in the raw stimuli. Since nodes have prototypes that represent features in the environment, a node can be used as a detector for the specific combination of features described by the node's prototype. The node would then produce an activation indicating the stimulus' similarity to the node's prototype, from 0.0 (totally dissimilar) to 1.0 (identical).

The SOM already uses a similarity function for training. This similarity function can be used to assign a node an activation $a = [\zeta_1,\zeta_2,...,\zeta_k]^T \in \Re^k$ using the Gaussian function a:

$$a_i = \exp\left(-\frac{\|x_i - m_i\|^2}{2\sigma^2}\right). \qquad 3.1$$

Formula 3.1 produces a value closer to 1.0 when the stimulus and the prototype are similar, and closer to 0.0 as the two grow further apart. The σ parameter can be used to control how fast activation drops off as the stimulus and the prototype grow more dissimilar. When a stimulus is presented to a SOM in CLA, Formula 3.1 is used to create the activation vector a. The activation vector is a new representation of the stimulus in terms of the prototypes m. Sometimes, a stimulus will activate a single node whose prototype represents the stimulus very accurately. Other times, a stimulus will activate multiple nodes more weakly, if no one node accurately represents the stimulus, but some nodes are similar to aspects of the stimulus. Together, the prototypes m along with its activation vector a are called a "CLA Layer."

The activation vector now can be used similar to the input vector that SOM uses as a stimulus. For this purpose, a new CLA layer is created whose prototypes have the same ordinality as the activation vector of the original layer. The activation vector then becomes a new stimulus that is input to this new higher layer. The higher layer can now organize prototypes from the activation vector of the lower layer. In this way, CLA layers can be stacked one on top of the other (Figure 3.1).

To train CLA, activation must be propagated from the bottom layer to the top layer before any training can happen. However, the necessary computation of comparing the stimulus to the prototypes is already done as a precursor to the training step. As a layer receives an input stimulus and scans the SOM for the winning node, all distances are converted to an activation using Formula 3.1 and stored. This activation process continues up the layers until they have all been activated.

At this point, the standard training described in Chapter 2 can take place. Each layer is trained on the activation vector of the lower layer, except for the lowest layer which trains on raw stimuli.

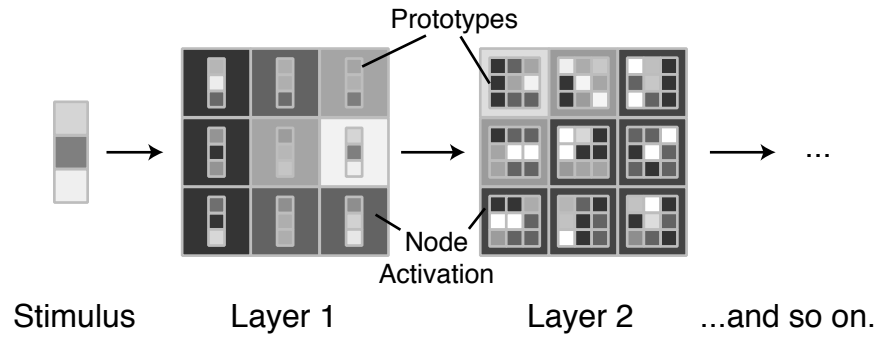

Figure 3.1: A CLA example. An example schematic of a simple CLA configuration. This CLA has two layers. The raw stimulus is presented to Layer 1 and compared to the prototypes in each node. Those with dissimilar prototypes have a low activation (indicated by a dark color in the node). Nodes with similar prototypes have a high activation (indicated by a light color). Activations of Layer 1 become the stimulus to Layer 2. Layer 2 prototypes are compared and the nodes are activates, just as in Level 1. Using the prototypes in a layer as feature detectors allows an arbitrary number of SOM layers to be connected into a hierarchy.

3.3 CLA Structures

Because the SOM has already projected the feature space onto a two-dimensional feature map, simply adding another layer on top of this will not learn anything new. Any additional layers will only re-represent the same information. However, CLA also supports more complex structures that introduce new information to each layer. Three examples of these structures — tree structure, time-delay structure, and recurrent structure — are described below.

CLA can organize layers in a tree structure, allowing a higher layer to receive activations from more than one lower layers. In this case, the higher layer's prototypes are a list of vectors, each matching the activation vector of each of the input layers. This technique can be used to integrate different sources of information, such as multi-model information. This structure is used in all of the experiments in the following three chapters.

CLA can also organize the layers to introduce a time delay between layers. With a time delay, the activation vector of a lower layer is held for one or more time steps before it is introduced to the higher layer. By itself, a time delay does not impact the system much, but the time delay can be combined with the tree structure to present a higher layer with the activation of a lower layer at multiple time steps. Such a technique can be used, for example, to create a "sliding window" of activation. This technique is used in the first application chapter, following this one.

CLA can also used the time delayed layer to build a recurrent layer structure. In this case, a higher layer's activation vector can be input to a lower layer so that the layer connections form a loop. This structure cannot be implemented without a time delay. The recurrent CLA structure can be used to build systems similar to Simple Recurrent Networks (section 4.5.1) that find structure in time. None of the experiments in this dissertation use a recurrent structure, but it is an important capability for CLA to support for future applications.

These are only three of a number of possible configurations of CLA. The right configuration will depend on the needs of the modeler. CLA is flexible enough to represent multiple levels of information using multiple information structures.

3.4 Fallback in CLA

While the previous sections describe how CLA builds a hierarchy of representations, addressing Information Processing Principles 2 and 3, this section describes how to address the functionality of principles 4 and 5, also known as fallback (described in section 1.1.2). According to the principles, fallback should occur when the highest level is overloaded, either by complex, unfamiliar or noisy information. Fallback is a useful technique that gives robots the ability to respond to changes in the environment or its own sensorimotor configuration by gracefully degrading in performance, rather than failing outright.

Implementing fallback in CLA is a simple procedure. CLA stores its knowledge in a hierarchy of layers, where each layer contains a set of prototypes. When CLA detects that the top layer or layers are overloaded, or that they otherwise cannot handle the stimulus, then CLA simply ignores the activation of the top layer and uses the activation of the next highest layer. The challenge, then, is to detect an overload and trigger the fallback mechanism.

With CLA, overload is defined to be the situation where the stimulus presented to a layer is underrepresented by the learned prototypes. In other words, a layer is overloaded when it receives a stimulus on which it has never been trained. This happens under three circumstances: (1) the layer is untrained, (2) the environment has changed or the agent has entered a new part of the environment with new information, or (3) the agent's sensory apparatus has been altered either through damage or — in the case of an infant — growth.

Keep in mind that a layer represents one level of prototypes, so a stimulus that is familiar for one layer may be novel for another layer. Using the typing fallback example from section 1.1.2, consider that CLA has been trained on typing with a QWERTY keyboard. CLA would learn, say, prototypes that describe how to press keyboard keys at one level, and prototypes describing the relative location of the keys at another higher level. If this trained CLA was then confronted with a Dvorak keyboard, the stimulus would be novel to the higher level, since all the keys are in a new place. The information about the relative key positions of the Dvorak keyboard would be underrepresented in the higher level. But the keys are still pressed in the same way, and so the prototypes at the lower level are still applicable. The multi-level representation of CLA is what allows fallback to work: when higher-level representations fail to represent the stimulus, then lower-level representations are used as a backup.

There are two techniques for detecting overload used in this dissertation. One way to detect overload is using a measure of total activation for a layer. The total activation would be the sum of the activation vector from a given layer. During training, each layer can store the minimum total activation that is generated from the stimuli. This measure is used as a threshold that indicates normal activation for a layer. When the total activation drops below this threshold, then this would indicate that the present stimulus is overloading the layer, and CLA should use a lower layer whose activation is above its threshold. This technique is good for propositional schemas, and is used in the experiments in Chapter 4.

An alternative method can be used if the learned schemas are sensorimotor schemas. In this case, there may be some indication as to the success or failure of the schemas that are activated. The reliability rate of the schemas are tracked and stored with the prototypes during normal operation. After training, CLA keeps track of the average reliability of all the sensorimotor schemas in a layer. If the average reliability of a layer's schemas drops below a certain percentage, this indicates

that the layer is overloaded, and a lower layer that is more reliable should be used. This technique is used in the experiments in Chapter 6, where it is also described in more detail.

These are just two techniques to detect an information overload and trigger the fallback effect, and there may be other techniques as well. CLA can support these and other approaches because CLA builds and maintains a hierarchical knowledge structure where any level of knowledge can be utilized when needed.

3.5 Conclusion

CLA is an implementation of the Information Processing Principles. It builds a hierarchical knowledge base from basic observations using interconnected SOMs communicate using activation vectors. CLA supports tree structures, time delay structures and recurrent structures. It also supports fallback, a useful technique for robust robotics.

The central thesis of this dissertation is that a model of infant cognitive development can give a robot the ability to learn grounded, robust control. CLA is such a model. The next chapter will demonstrate that CLA models infant cognitive development. Chapter 5 demonstrates that CLA can learn sensorimotor schemas in addition to propositional schemas. And Chapter 6 shows that CLA can be used as a grounded, robust controller for a mobile robot.

4. Cognitive Development Model

The previous chapter introduced CLA, an adaptive robot controller that provides robust autonomy by modeling infant cognitive development. The central thesis of this dissertation is that a constructivist model of infant cognition can be used to provide an autonomous agent with robust control. In this chapter, we demonstrate CLA's ability to model infant cognition by using CLA to replicate a set of studies in the acquisition of causal perception, an important and challenging area in cognition. In the following chapters, CLA will be applied to autonomous robotics (Chapter 5) and learning with delayed rewards (Chapter 6).

The goal of the experiments described in this chapter is to demonstrate that CLA can accurately model infant cognitive development. Specifically, CLA learns to identify billiard ball launching events as causal events. It does this using constructivist learning, starting with a simple component view of the events and building a higher-level causal view. Moreover, CLA demonstrates its ability to degrade gracefully. When presented with a complex event, CLA does not fail completely, but falls back to a simpler representation. Fallback, the ability to drop back to a lower level of knowledge when the higher levels fail (section 2.1.3), is an important aspect of constructivism, and specifically the Information Processing Principles (Cohen, Chaput & Cashon 2002). Fallback is used later in Chapter 6 for robust control. Following the experiments, a body of similar work in models of infant cognitive development is reviewed.

4.1 Causal Perception

Causality is an excellent domain to demonstrate CLA as a cognitive model. First, there is a wealth of data on this subject in experimental psychology, particularly many recent studies in infant acquisition of causal understanding. Also, it is a complex, time-oriented domain that pushes the limits of any learning system. Causal events are not simply images or snapshots of images at one time, but are continuous and take place over many steps. Above all, causality is an ancient problem that has been at the center of modern philosophical debate since Hume (1777/1993), who's "blank slate" approach proposed that all knowledge is learned, and Kant (1794/1982), who insisted that certain concepts, particularly causality, could not be learned and therefore must be innate. A model of causality using CLA is not only a rigorous test of a learning system, it could also make a valuable contribution to a centuries-old psychological and philosophical debate.

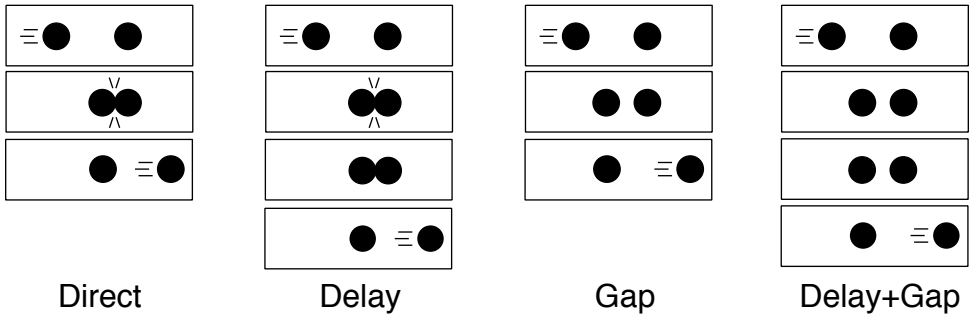

Figure 4.1: Launching events. Schematics of the four different launching events used by Michotte (1963) to study causal perception in adults. The Direct event is the normal occurrence when one billiard ball hits another. The Delay event introduces a delay at the moment of collision. The Gap event has the balls "collide" when they are still apart. The Delay+Gap event has the combined features of the Delay and Gap events. These same events are used in habituation studies to detect causal perception in infants.

4.1.1 Causal Perception in Adults

This experiment, like many contemporary studies of causality (described in the next subsection), uses a set of events first developed by Michotte (1963). Michotte explored causal perception in adults using simplified billiard-ball collisions, called launching events (see Figure 4.1 for a schematic of these events). Michotte found that adults presented with a simple direct launching event would describe the event as causal. He could then alter the likelihood that subjects would identify the event as causal by altering two components of the event. He could either introduce a delay at the moment of the collision, or a gap between the two balls at the point of collision, or both. He found that manipulating the temporal component of the event (increasing the delay) or the spatial component of the event (increasing the gap) reduced the likelihood that the subjects would identify the event as causal.

4.1.2 Causal Perception in Infants

Leslie (1984) used this same approach to identify causal perception in infants. He used a habituation paradigm to determine whether infants were attending to the spatial and temporal components of the events, or the causality of the events. Leslie placed the four events on a theoretical 2×2 grid (Figure 4.2) and tested infants using a habituation study that compared events across the diagonals of the grid. In other words, some infants were habituated to a Direct event and tested on a Delay+Gap event (or vice versa). Other infants were habituated to a Delay event and tested on a Gap event (or vice versa). Leslie reasoned that, if infants had a component view of the events, and they were just responding perceptually in terms of the spatial and temporal properties of the events, then dishabituation along one diagonal should be similar to dishabituation along another diagonal, because both pairs of events involve the same change in both components. However, if infants had a causal view of the events, then dishabituation along the Direct-to-Delay+Gap diagonal should be greater than along the Delay-to-Gap diagonal, because only the Direct event is causal. Leslie found that 6.5-month-old infants did, in fact, respond to the events in accordance with the causal view.

Further, Cohen and Amsel (1998) used a similar paradigm and found interesting developmental changes in infants' responses to causality. Rather than test across the diagonals, Cohen and Amsel compared dishabituation between Gap and Direct events with dishabituation between Gap

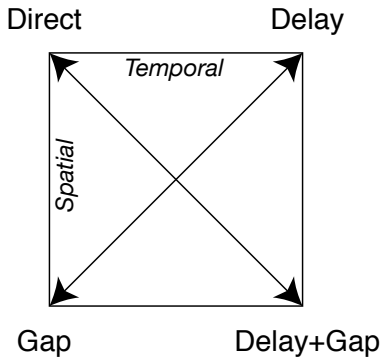

Figure 4.2: Launching events viewed as components. The four launching events placed on two axes: a spatial axis representing a difference in spatial features, and a temporal axis representing a difference in temporal features. Infants using the component view should dishabituate equally along the diagonals, while infants using a causal view should dishabituate more between Direct and. non-Direct events than between two non-Direct events.

and Delay events. If infants were responding in terms of causality they should have dishabituated more when going from the Gap to Direct event, because that difference includes a change in causality. Cohen and Amsel found that 6.25-month-olds did respond to the events causally, thus replicating Leslie. However, they also found that 5.5- and 4-month-olds responded to the events in a way more consistent with a component view. Four-month-old infants were sensitive mainly to the duration of movement in the event, whereas 5.5-month-old infants were also sensitive to the spatial and temporal components. Neither age responded in terms of the causality. These results provide an indication of a developmental shift in causal understanding that progresses from a component view to a higher-level causal view.

4.2 Modeling Causal Acquisition with CLA

While the studies described above suggest a simple progression from one stage to another, the Information Processing Principles hypothesize a relationship between the component view and the causal view. These stages do not occur sequentially and independent of one another. Rather, the component view is used to build the causal view. The challenge of a constructivist model of causal perception is not just to learn causality, but to learn causality from its components.

To model the part-to-whole progression of causal acquisition, CLA is designed to first learn a component view of causal events and then to use this knowledge to learn causality. Cohen and Amsel (1998) demonstrated that younger infants see launching events as components so, accordingly, the model is trained on the spatial and temporal components of launching events. The two different components are presented to CLA simultaneously using two separate representations. See Figure 4.3 for a diagram of example inputs.

The first representation, the Position Input, captures the spatial information of the event and excludes any temporal information. At each time step of the launching events, the position of each of the two balls are represented on a 20-element vector. A ball starting at the far left would be represented by the first element being set to 1.0, while another ball waiting to be struck at the center of the screen would be represented by the 11th element being set to 1.0. There are always

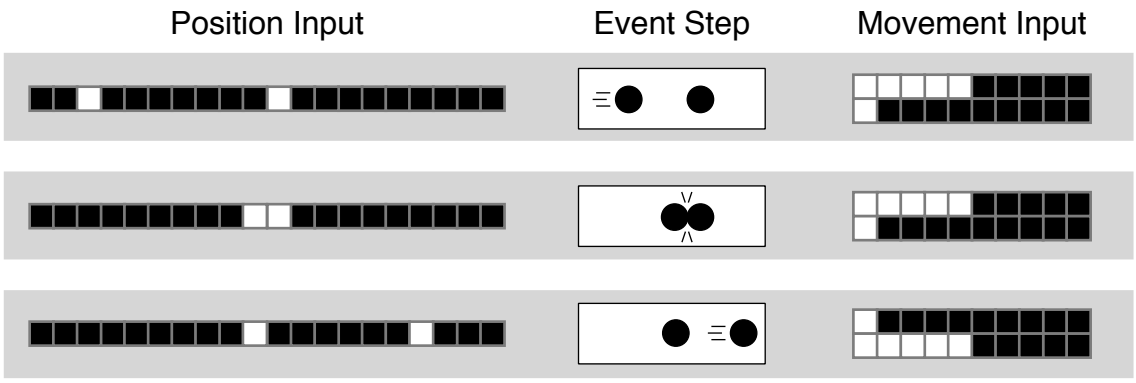

Figure 4.3: Launching event input. The Position and Movement Input for three steps in a Direct launching event. Each element is either 0.0 (black) or 1.0 (white). The Position input represents the position of the balls, but not the temporal aspects. The Movement Input represents the speed of the ball, but not the spatial aspects. A sliding window of three consecutive Movement Inputs are presented to CLA, which can then capture the change in movement (acceleration). They are presented to the learning system separately to address the challenge of learning the causal view from the components of the launching events.

2 elements in this vector set to 1.0, while the rest are set to 0.0. These positions are presented as snapshots to the learning system.

The second representation, the Speed Input, captures the temporal information of the event and excludes any spatial information. The speed of each ball is represented with a 10-element vector acting as a speedometer for each ball. When a ball is still, the first element is set to 1.0 while all other elements are set to 0.0. As the ball accelerates, more elements are set to 1.0 starting with the second, then the third, and so on. Each ball's speed is represented by its own vector so that both balls are represented with a 2-by-10 Speed Matrix. Moreover, to further capture the temporal elements of this event, a sliding window of three consecutive snapshots of the Speed Matrix are collected and used as the final Speed Input.

The first experiment, a model of the acquisition of causal perception, is not meant to suggest that infants process causal events with such a stark and austere separation between the spatial and temporal components. However, we do know that infants have access to this information and process them separately to some degree. They are strongly separated in this experiment to test the hypothesis that the causal event can be learned from the components. If anything, this extreme separation should make learning causality more difficult.

The inputs are presented simultaneously to CLA, as illustrated in Figure 4.4. Each separate representation is presented to a separate SOM layer simultaneously. The Position Input is presented to the Position Layer, and the Movement Input is presented to the Movement Layer. Collectively, these layers comprise Level 1, which should learn a component view of the events. These second-level layers come to represent, as prototypes, the different positions and speeds over the course of an entire event. So, for example, different regions of the Position Layer would represent the balls being far apart, close together, or touching. Different regions of the Movement Layer would represent both balls stationary, or only the first ball moving, or only the second ball moving.

The Top Layer of the model receives inputs from both the Position and Movement layers. It will represent the event as a whole and, after training, distinguish causal from non-causal events.

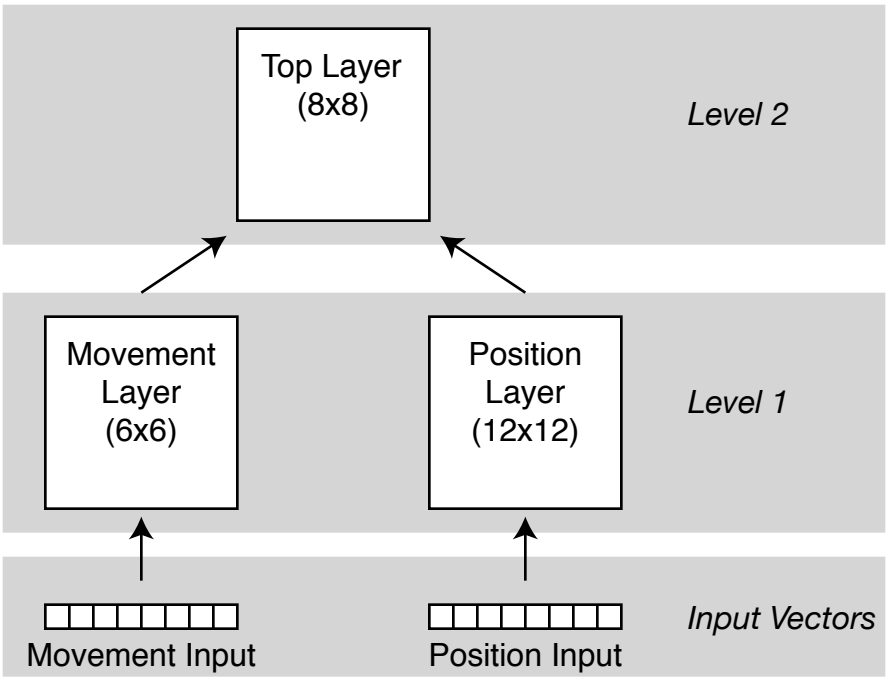

Figure 4.4: CLA design for learning causality. The Movement Input and Position Input are presented to the Movement and Position layers, respectively. The activations from these layers in Level 1 are then presented to the Top Layer in Level 2. Level 1 is expected to develop a component view of the launching event, while Level 2 is expected to develop a causal view.

4.3 Experiment 1: Acquisition of Causal Perception

The first experiment uses the inputs and architecture described above to replicate the results from the studies of infant perception and acquisition of causal perception by Leslie (1984) and Cohen & Amsel (1998). In doing do, CLA is demonstrated to model cognitive development in infants. More specifically, CLA is shown not only to learn causal perception, but to learn causal perception in the same way that infants learn it: by constructivist learning. CLA's hierarchical knowledge base will be used to demonstrate fallback in the next experiment, and operate as a robust controller of robots in the following two chapters.

4.3.1 Training Parameters and Measurement

During training, each event was presented to the model at a frequency using a gross estimate of its relative frequency in nature. Based on these estimates, Direct events were presented 85% of the time and each of the other events (Delay, Gap, and Delay+Gap) were presented 5% of the time. It is postulated that the frequency of experiencing these events, while certainly not the only information used by infants, plays a crucial role in the acquisition of causal understanding. If the world consisted mostly of Delay events, for example, infants would develop a radically different view of causality.

4.3.2 Level 1: Component View

As each layer received more training, different prototypes in each layer came to represent

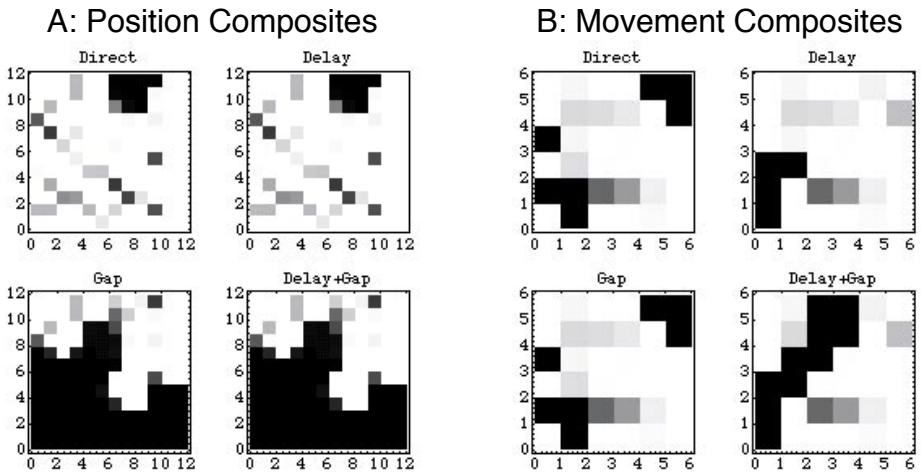

Figure 4.5: Activation composites of Level 1 layers for all four events. Each graph shows the composite activation across a launching event. The composite is the sum of all activation at every node. The composites for the Position Layer (A) is on the left, and the Movement Layer (B) is on the right. For each of these, the graph represents, starting with the top left and moving clockwise, the Direct, Delay, Delay+Gap and Gap events. These composites show that the Level 1 layers have learned the components of the launching events. (There is a section of the Delay nodes that does not activate for the Delay+Gap event, although the active nodes are clearly a subset of the active nodes for the Delay event. A possible explanation for the different is discussed below in section 4.3.5, "A Causal Continuum.")

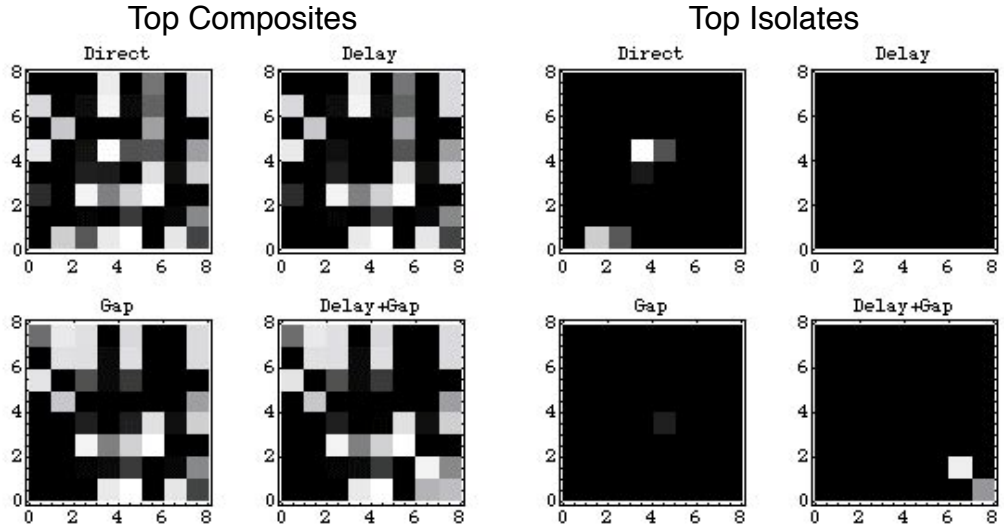

Figure 4.6: Activation composites and isolates for the Top Layer. The graphs on the left show the composite activation at the Top Layer for the Direct, Delay, Delay+Gap and Gap events. There are differences in both axes, indicating that each event has its own representation. The four graphs on the right show the activation isolates — nodes that are active only for a single event — at the Top Layer for the Direct, Delay, Delay+Gap and Gap events. There are nodes that only activate for the Direct event, but there are no such nodes for the Gap or Delay events, demonstrating that the Top Layer has a special representation for the Direct event that doesn't exist in the lower levels. (See section 4.5.3 for a discussion of the specialized nodes in the Delay+Gap event.)

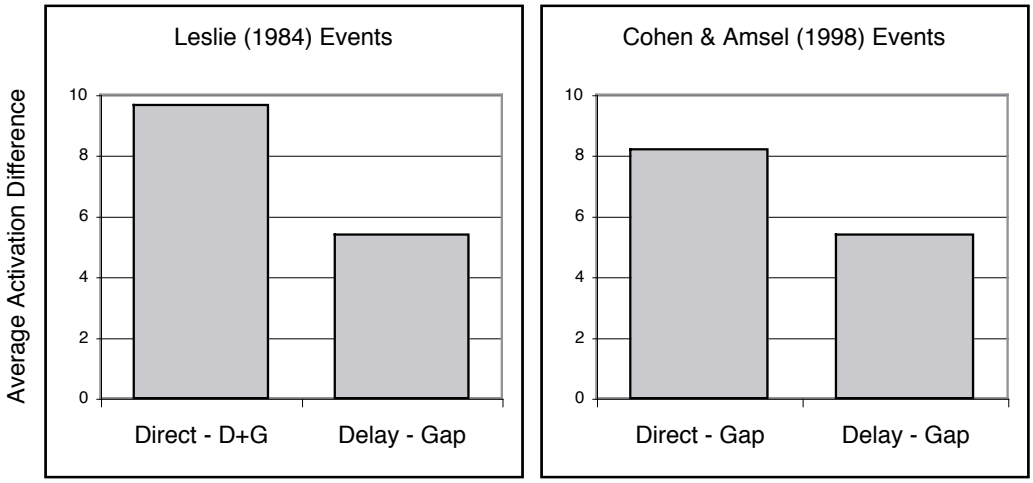

Figure 4.7: Difference in total activation at the Top Layer between events. Each bar represents the difference in the total activation of the Top Layer across an entire event. On the left, the event pairs are taken from Leslie (1984): Direct to Delay+Gap (a causal difference) and Delay-Gap (no causal difference). On the right, the event pairs are from Cohen and Amsel (1998): Direct-Delay (a causal difference) and Delay-Gap (no causal difference). In both cases, the change in causality resulted in a significantly higher difference, just as with infants.

the launching events at different points in time. If all of the activations from all the time steps are added together, we get a composite activation graph showing all the prototypes that were activated during the event. Figure 4.5A shows the composite activation in the Position Layer for all four events. Although there is a clear distinction in the activation patterns between events that have a gap and events with no gap, there is no distinction between Delay events and non-Delay events. This activation pattern reflects that certain prototypes have specialized to represent the presence or absence of a Gap, but not of a Delay. Conversely, Figure 4.5B shows the exact opposite pattern for events in the Movement Layer. There are specialized prototypes for the presence and absence of a Delay, but not a Gap. The lower-level layers are working as they were designed, reflecting the components of the event.

4.3.3 Level 2: Causal View

The composite activation graphs for the Top Layer in Figure 4.6 show no such symmetry. While the lower levels had prototypes that specialized for the components of the launching events, the Top Layer has prototypes that specialize for the events themselves. To identify which prototypes have specialized for which events, we can generate an "isolate" graph by taking the composite for a given event and subtracting out the composites for the other three events. (This procedure is similar to the process used when making an fMRI; Horowitz 1995.) The results of this subtraction can also be seen in Figure 4.6. As the figure shows, there are specialized prototypes that activate exclusively during a Direct event, although there are no such specialized prototypes for Delay and Gap events. The presence of specialized prototypes shows that the Top Layer has created a unique representation for the causal event that does not exist for other non-causal events. The specialized prototypes that appear for the Delay + Gap event also fit the predictions of Leslie (1984) in an unexpected way and will be discussed in more detail in section 4.3.5.

Figure 4.8: Spatiotemporal continuity of launching events. Taken from Leslie and Keeble (1987), who proposed this causal view of the launching events. As opposed to Figure 4.2, where events are compared by their components, the spatiotemporal continuity has a single dimension: causality. They posited that infants with a causal view would respond to events using this continuity, responding only to differences in the causality of the event. (Leslie and Keeble offered no evidence or rationale for the order of the Gap and Delay events in the middle, and so the order is assumed to be arbitrary.) Without setting out to do so, CLA has reproduced this phenomenon.

4.3.4 Comparison to Infant Experiments

To compare these results with the average infant looking times found by Leslie (1984) and Cohen and Amsel (1998), the looking times are compared to the difference in activation between pairs of events. To get the difference in activation, the composite of one event is subtracted from the composite of another event, and the resulting differences in activation are then added together. This procedure gives the sum value of all the node activations that occur in one event and not another and gives us a meaningful measure of novelty. The results of these comparisons for the events used by Leslie (1984) can be seen in Figure 4.7. The difference in activation is significantly greater between the causal (Direct) and non-causal (Delay+Gap) events (M = 9.66, S.D. = 1.16) than between the two non-causal events (Delay and Gap) (M = 5.38, S.D. = 1.00), t(6) = 5.58, P = .002 (two-tailed), d = 4.28.

This pattern of results is consistent with the difference in looking times found by Leslie, who considered his results to be evidence for infant causal perception. A similar pattern of results was found by Cohen and Amsel (1998), who also took their findings as evidence for infant causal perception. Again, our model produced activation levels consistent with the empirical data. As seen in Figure 4.7, the activation difference is significantly greater between the causal (Direct) and non-causal (Gap) events (M = 8.20, S.D. = 0.56) than between the two non-causal events (Delay and Gap) (M = 5.38, S.D. = 1.00), t(6) = 4.93, P = .005 (two-tailed), d = 2.82.

4.3.5 A Causal Continuum

The model's treatment of the Delay+Gap event was an unexpected surprise. Rather than just building specialized prototypes for the Direct event, the Top Layer also built specialized prototypes for the Delay+Gap event (Figure 4.7). In effect, the model did not just represent events in terms of "causal" or not, but on a continuum from "causal" to "less causal" to "not causal." The layer nodes that exclusively represent the Delay+Gap event allow the system to view these events as *less causal* then an event with just a Delay or Gap by itself. Such a continuum was first proposed in adults by Michotte (1963) and in infants by Leslie and Keeble (1987). Figure 4.8 is a reproduction of Leslie and Keeble's (1987) "spatiotemporal continuity." It is wholly compatible with CLA's results, placing the Direct and Delay+Gap events at the extremes, while placing the Delay and Gap events in the middle.

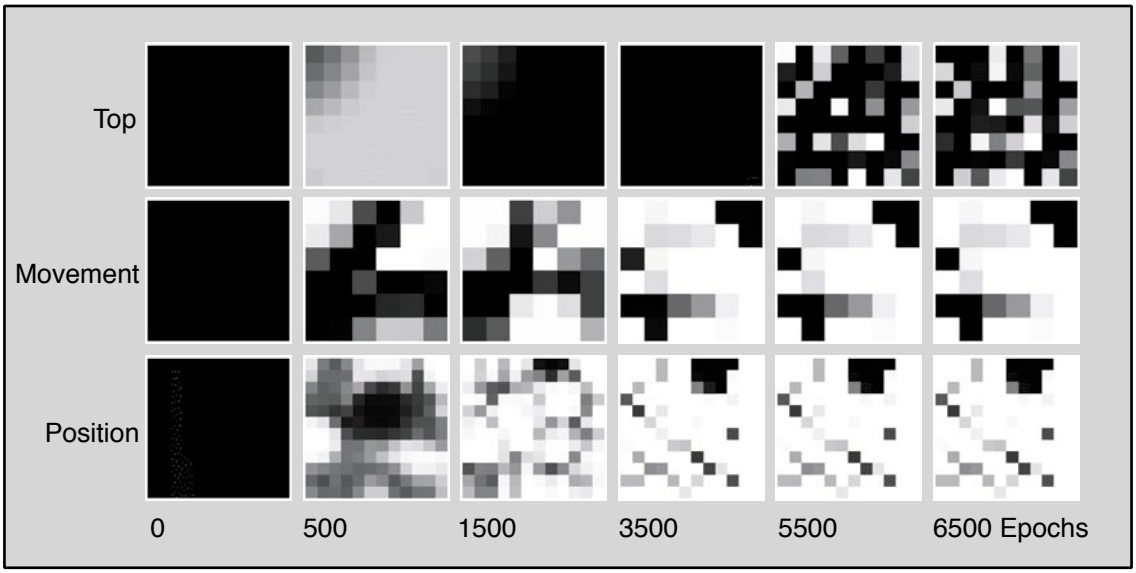

Figure 4.9: Stage-like development of causality. Composite activation of the Position, Movement and Top layers for a Direct event at different points in training. While the lower layers begin to settle on an activation pattern at around 4500 training epochs, it is not until this point that the top layer begins to form any representations. This demonstrates that, even through all layers use the same learning algorithm, higher levels do not form representations until the lower levels have settled on a representation. This stage-like development was first proposed by Piaget.

4.3.6 Stage-Like Development

The model also provides evidence of stage-like development. In Figure 4.9, we can see the composite activation for the Direct event in the three layers at different points in training. Although the lower Position and Movement Layers begin to form immediately, and start to settle at about 4500 epochs, it is not until this point that the Top Layer begins to form. This is a consequence of CLA's hierarchical design. Higher layers are organizing patterns of activity in lower layers, but they cannot organize anything consistently until the lower layers are stable. Even though CLA uses a continuous learning system, the development progresses in stages from one level (the component view) to the next level (the causal view). Stage-like cognitive development in infants was first proposed by Piaget (1937) and is consistent with Cohen and Amsel's (1998) results that infants at different ages process the same events differently.

4.3.7 Conclusion

In this experiment, CLA learned a causal view of launching events. It learned this view starting with a component view and integrating the components in a higher CLA layer. The progression from a component view to a causal view replicates the results of studies by Leslie (1984) and Cohen & Amsel (1998) demonstrating the presence and acquisition of causality by infants. The events learned by CLA lie along a Causal Continuum, which is consistent with the results of Leslie and Keeble (1987). And CLA learns causality in stages, not all at once, which is also consistent with Piaget's (1998) observation of stage-like development in infant cognition.

In summary, this experiment demonstrated that CLA builds a hierarchy of knowledge in the same way as infants. The Information Processing Principles state that such a hierarchy should also be able to fall back, meaning it should respond to information overload at the top levels by utilizing layers at lower levels. CLA is shown to have this property in the next experiment.

4.4 Experiment 2: Fallback During Overload

While the previous experiment demonstrated that CLA models constructivist learning, it is also important that CLA can handle noise and overload robustly using fallback (section 1.1.2). Recall that fallback is described by Information Processing Principles 4 & 5: The learning system prefers to process information at the highest level, but should higher schemas be unavailable, lower schemas are still be available to process the stimulus. Fallback is important because it is what allows CLA to operate as a robust controller for mobile robots, as demonstrated in Chapter 6.

In section 1.1.2, fallback was illustrated by noting that an experienced typist, when confronted with a different keyboard layout, will regress to an earlier skill level but eventually recover full typing skills. Cohen and Oakes (1993) demonstrated the fallback phenomenon in infants. They found that 10-month-olds, who had no problem viewing causal events when the objects were simple toy cars, could not process causal events in the same way if the toy cars changed from trial to trial during habituation. Even though the infants were receiving the same spatial and temporal information, 10-month-olds processed these events with a component view, just as 4-month-olds process causal events. It was as if the 10-month-olds had regressed to an earlier stage of causal perception. As they grew older, however, infants eventually began to process these events as causal. Cohen and Oakes concluded that the changing agents in the launching events had introduced a categorization task, making it difficult for the infants. Eventually, however, the infants were able to integrate the more complicated information into their higher-level view.

In this experiment, CLA demonstrates that it not only learns like infants, but it fails like them too. When presented with a stimulus that is noisy or complex, CLA will still be able to process the event at a lower level. CLA has the ability to fall back because the higher-level learning is not destructive to the lower levels. The lower levels remain in place to be utilized should the higher levels fail.

4.4.1 Input Vectors for The Noisy Event

To see evidence for fallback, CLA is tested on a Noisy launching event, in which the speed varied throughout the event, and the position of the moving balls move forward irregularly, as if the balls were elliptical. This is a novel event for the trained model. It bears some resemblance to a direct causal event, but there is enough variation to make it difficult to identify as such. It is hypothesized that the lower level should receive some activation, while the higher level's activation should be muted.

The Noisy event was presented to a CLA trained in the method described above. Composite activation matrices were created for the Position, Movement and Top layers for the Noisy event.

4.4.2 Network Response

The composite matrices are shown in Figure 4.10. The lower level layers largely respond to the Noisy event as if it were a direct event, although there are some notable gaps in the activation. However, the Top layer's activation is greatly diminished, indicating that there is little response at this higher level. A system that used these activations would fail to recognize this event as a causal event, but would still be able to respond to this event using the components of the event.

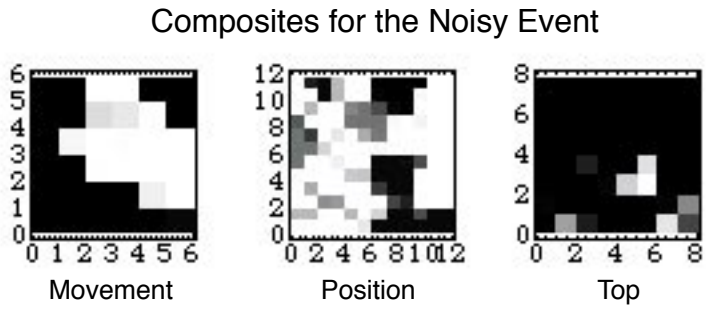

Figure 4.10: Activation composites for the Noisy event. Graphs of the activation composite matrices for over the Noisy launching event in the Movement, Position and Top layers. While the lower layers can largely recognise the Noisy event as a launching event, the Top layer has very little activation. The Noisy event is difficult to recognize as a causal event, but the CLA still responds to the event's spatial and temporal components.

4.4.3 Experiment 2 Discussion

CLA was shown to exhibit the ability to fall back when presented with a Noisy event. These results are consistent with Cohen and Oakes (1993), and further demonstrate that CLA is a model of infant cognitive development and processing. From a robotics point of view, these results show that CLA will not fail completely when confronted with a change in the environment, but will only suffer a partial regression and degrade gracefully (like the typist in section 1.1.2).

4.5 Related Work

Computational models of cognitive development have become a cottage industry in psychology, and they are having an effect on the study of cognitive development (see Elman et al 1996). This section reviews contemporary research in this area, with a comparison to CLA, the Information Processing Principles, and the work presented in this chapter. The majority of these models use some variant of the feedforward neural network, which is covered first. Other approaches are discussed as well, including self-organizing models, reinforcement learning, and the schema mechanism.

4.5.1 Feedforward Neural Networks

Feedforward neural networks are by far the most popular tool for contemporary models of cognitive development, driven largely by the appeal of a physiologically based system being used to produce (and reproduce) psychological phenomena. Feedforward networks use interconnected nodes to process sensory information. The sense vector is presented to the system as a set of input nodes. These nodes are connected by a set of weights to a layer of hidden nodes. Finally, the hidden layer is connected via another set of weights to an output layer. By processing input through two layers of weights, the system can produce non-linear output.

There are several features common to all the feedforward models discussed in this section that set it apart from CLA. First, they are all supervised learning systems. In order for any feedforward network to learn, the learning system must use corrective feedback or an error signal. This error signal is used to adjust the weights of the network so that an input will produce the appropriate output. For many domains of cognitive development, the source of an error signal in the environment is not entirely clear, causing this approach to be controversial. CLA, on the other

hand, is an unsupervised learning system. CLA does not use an error signal, but categorizes all of the inputs against each other using only their features.

Second, all the systems below use the backpropagation learning algorithm for multi-layer neural networks. Training multi-layer networks is not obvious, and the method for doing so was not available until the mid 1980s. The backpropagation learning algorithm solves the weight setting problem mathematically, but it is agreed that this learning algorithm is not an accurate model of how learning actually occurs in the brain. While neural plausibility is not crucial to this thesis, it is important to infant cognition researchers, making the backpropagation algorithm an Achilles' Heel to many of these models. In contrast, CLA uses the SOM, which has been shown as an excellent model of neural processing (see Kohonen 1997).

Finally, none of the models in this section use a hierarchical learning system. According to the Information Processing Principles, such a system is necessary for constructivist learning. The reason that there are no hierarchical feedforward models of infant cognition could be that it is not evident how one would make such a model using standard feedforward networks. CLA is a hierarchical learning system that can model the part-to-whole process described by constructivist learning.

With these three main differences in mind, this section will discuss the different variants of the feedforward network in more detail below.

Backpropagation Networks

Feedforward networks first gained wide acceptance in infant cognition research with the publication of the watershed two-volume PDP compendium (Rumelhart, McClelland et al 1986; McClelland, Rumelhart et al 1986). A relevant article within this collection contained a backpropagation network model of infants' acquisition of past tense endings (Rumelhart & McClelland 1986). The system was trained to produce the appropriate past tense ending for a presented word, and the system's performance resembled infants' performance at this task: it performed fair, then it was terrible, then it was very good. (This is referred to as the U-Shaped Curve.)

As mentioned earlier, it is not always clear where the error signal for supervised learning would come from. However, for some domains — most famously, language — it has been shown that any error signal that might be present in the environment would not be sufficient to learn in the domain (Chomsky 1968), otherwise known as the poverty of the stimulus. Thus, the Rumelhart & McClelland (1986) study was criticized for "giving away the answer" by providing normative feedback that didn't exist in an infant's environment. This is a common criticism for infant models that use feedforward networks and, consequently, unmodified backpropagation networks are seldom used anymore. When they are (as in Mareschal & Johnson 2002) they fall under the same criticism (Cohen & Chaput 2002; Marcus 2002; Munakata & Stedron 2002; Smith 2002).

Supervised learning is not an issue for CLA. CLA uses the SOM, which is trained using an unsupervised learning system. Like the SOM, CLA does not use an error signal to build representations. CLA is a self-organizing learning system that builds knowledge by relating inputs to each other.

Simple Recurrent Networks

Elman (1990) proposed a more plausible approach to error-based learning for finding structure in time-based or sequential information such as language. Elman proposed that the system should learn, given a stream of words, to predict the next word in the sequence. To allow the network to learn using context, Elman used a Simple Recurrent Network (SRN) that combines the input of the current word with the activation of the hidden layer from the previous word. SRNs are still widely used in machine learning, and are also used to model infant cognition (Lewis & Elman 2001).

This approach to error-based learning is more plausible from an environmental perspective, since infants clearly do have access to the next item in some sequence (like the next word). However, these SRNs still rely on the controversial backpropagation learning algorithm, and do not use a hierarchical approach.

Auto-associator Networks

Another approach to error-driven learning is the auto-associator network (Labiouse & French 2001). This network works just like a backpropagation network, except that the network is trained to produce output that is identical to the input. Like other forms of dimensionality reduction, auto-associator networks learn a set of weights that capture the features of the stimuli. There is no poverty of the stimulus argument for auto-associator networks, since the error signal comes from the same source as the input itself. In fact, this approach to learning is similar to SOM and other self-organization learning systems. Auto-associator networks have been used to model a number of domains in infant cognitive development, including language and grammar learning (Sirois et al 2000; Sirois 2004), categorization (Mareschal et al 2002), and visual object processing (Westermann & Mareschal 2004).

While auto-associator networks avoid error-driven learning, they are highly implausible models of information processing in infants. Unlike the SOM's approach to self-organization, which is affirmed by a body of neuroscientific research (Kohonen 1997), there is no such affirmation for auto-association. Biological plausibility aside, using a self-organizing system addresses the issue of learning without relying on environmental feedback. Still, auto-associator networks are seldom used to learn hierarchical knowledge structures.

Cascade-correlation

A variation of the auto-associator network uses cascade-correlation (Fahlman & Lebiere 1990) as a learning system. Cascade-correlation operates as a standard backpropagation network during learning, using the input as the target output. Once the error between the input and output has been minimized, the system introduces a new node to the hidden layer, and the error is minimized once again. The idea is that the initial set of hidden nodes will develop weights that categorize a the most dominant features in the input set, but there will be "exceptions," or input vectors that are not properly reproduced using the initial hidden node weights. When a new node is introduced to the hidden layer, this node learns to reproduce some of the input vectors not previously learned, leaving only a second set of exceptions. More nodes are added until all the exceptions are handled.

Cascade-correlation networks have been used to model a wide variety of infant cognitive development (Shultz & Bale 2001; Shultz & Rivest 2001; see Shultz 2003 for an overview). This

learning system has also been suggested as appropriate for higher-level constructivist learning because it can grow dynamically (Shultz & Mareschal 1997).

Still, cascade-correlation does not use a hierarchical knowledge structure like CLA. While cascade-correlation adds new nodes to learn new regularities, the new nodes still learn representations that operate at the same level as the original hidden nodes. In other words, no matter how many hidden nodes are added. Cascade-correlation has not been shown to develop the higher-level representations of other hierarchical learning systems like CLA. CLA does not learn to simply handle exceptions; it learns a new set of knowledge based on regularities in the knowledge of lower layers. CLA's ability to build entirely new knowledge allows it to represent a stimulus at multiple levels simultaneously, and giving it the high-level processing preference and ability to fall back as described in the Information Processing Principles.

4.5.2 Reinforcement Learning

Schlesinger & Barto (1999) used Reinforcement Learning to model the development of causal perception. Reinforcement Learning uses a reinforcement signal to associate states with actions that will result in the highest reward. In this case, the simulated agent received reinforcement when it would move its simulated eye to follow a moving billiard ball. The ball would move behind an occluding screen and reappear on the other side. The system eventually learned to anticipate the ball by moving the eye to look at the far edge of the screen before the ball emerged.

While Schlesinger and Barto demonstrate that reinforcement learning can learn this behavior, it is not very convincing as a model of cognitive development. Most obviously, there is no explanation for why a child would receive a reward for looking at a billiard ball. But a deeper problem is that the system doesn't actually demonstrate development of any kind, only the end result. CLA replicates the infant causality studies not only at their endpoints, but during learning as well.

4.5.3 Self-Organizing Models

CLA uses the SOM as a foundation work, and the SOM has modeled a great number of cognitive activities, including vision, audition, speech, and kinetics (Kohonen 1997). Kohonen (1988) even suggests methods for using the SOM for habituation and familiarization. However, the original SOM is rarely used as a model of infant cognitive development.

However, recent work in cognitive development models has been done with learning systems called Hebbian systems. Munakata and McClelland (2003) model grammar learning with these systems. Munakata (2003) also shows that these systems resemble neural systems in the brain. The kind of Hebbian networks studied by Munakata & McClelland are unsupervised self-organizing learning systems, and Munakata & Pfaffly (2004) include the SOM as a learning system that accurately captures their approach to Hebbian learning. Like the SOM, these systems do not learn hierarchical data structures, while CLA builds a hierarchical knowledge base.

Bednar and Miikkulainen (2000a & 2000b) used a variant of the SOM called HLISSOM (Bednar & Miikkulainen 2001) to model innate face visual preferences in infants. HLISSOM's hierarchically arranged self-organizing maps is similar to CLA. However, the goal of Bednar and Miikkulainen is a rigorous neural model based on neuroscientific research of the visual pathways in the brain. Thus, each layer and its role is predetermined before learning begins. CLA does not

predetermine the role of each layer, but allows each layer to learn whatever regularities exist in the layers below.

Farkaš & Li (2001) use a SOM to organize the hidden unit activations of a SRN to learn the meanings of words. The model successfully grouped grammatical and semantic categories based on their placement in training sentences. This system is a clever combination of different learning systems for their different strengths. Like CLA, it is self-organizing and hierarchical. However, this system also has predetermined the roles of the different layers ahead of time. Additionally, while CLA is a homogeneous system of SOMs, Farkaš & Li use SOMs and SRNs, so it is not readily apparent how to build this system to include more levels.

Finally, one fascinating application uses self-organizing learning systems to model mother-child interaction in order to mechanically generate infant speech (Yoshikawa et al 2003). The simulated infant has a SOM that organizes perceived auditory information from its mother and itself. These categories form the basis of a vowel sound production process that attempts to mimic the sounds that come from the mother. Like Farkaš & Li's system above, this system builds knowledge at multiple levels, but again these levels are predetermined and the system is not designed for an arbitrary number of levels.

4.5.4 Evolutionary Systems

Evolutionary systems have also been proposed as a mechanism for modeling infant cognitive development (Schlesinger & Parisi 2001; Schlesinger 2004). Evolutionary systems use a genome to describe an agent's behavior. During training, multiple genomes are created and tested for fitness, usually by how well they perform a given task. The strongest genomes are then evolved through mutation and cross-breeding. Over time, a genome is developed that performs well at the task.

Putting aside the improbability of infants using evolution to learn during their first years, these studies never actually apply evolution to cognitive development. Perhaps when these systems actually model infant cognitive development there will be a basis for comparison to CLA.

4.6 Conclusion

CLA was shown in this chapter to model infant cognitive development. Specifically, CLA reproduced contemporary studies of causal acquisition in infants. CLA not only learns to view events as causal, but learns in the same way that infants learn. Additionally, CLA also replicates infants' ability to fall back to lower levels of representation when the stimulus is complicated or noisy.

CLA captures all of the elements of constructivist learning as delineated by the Information Processing Principles (Cohen, Chaput & Cashon 2002), allowing CLA to operate as a robust controller for mobile robots. The schemas learned in this experiment, though have been propositional schemas that classify observations. To control a robot CLA must build sensorimotor schemas that relate observations with actions (section 2.1.1). CLA's ability to build sensorimotor schemas is shown in the next chapter.

5. CLA for Mobile Robots

In the previous chapter, CLA reproduced infants' acquisition of causal perception, demonstrating that CLA is a model of cognitive development. The acquisition of causal perception mirrored the constructivist learning process used by infants, and supported fallback when exposed to confusing stimuli. However, modeling cognitive development and robot control are two different things. In the experiments in the previous chapter, CLA built propositional schemas (section 2.1.1) that classified and related sensory input. For CLA to control a robot, CLA must also be able to learn sensorimotor schemas that relate sensory information with actions. In this chapter, we apply CLA to a simple agent model to demonstrate that CLA is not limited to cognitive models or propositional schemas but can also build sensorimotor schemas and thus be applied to autonomous agents. CLA's ability to learn sensorimotor schemas will lead, in the next chapter, to a demonstration of a real robot using CLA to learn a goal-directed policy from delayed reward.

To demonstrate that CLA can learn sensorimotor schemas, CLA is used to implement the Schema Mechanism (Drescher 1991), a landmark model of robot learning which builds sensorimotor schemas using a constructivist approach, but which has some shortcomings that prevent its application to real-world robots. CLA not only replicates all of the functionality of the Schema Mechanism, but does so more efficiently.

The first section of this chapter gives an overview of the Schema Mechanism and reviews its application to the Microworld, a very simple agent and environment. The next section compares the Schema Mechanism to CLA, discusses some of the efficiency problems of the Schema Mechanism, and describes how CLA addresses those problems. The third section describes the details of how CLA is used to reimplement the Schema Mechanism. In the fourth section, CLA is applied to the Microworld, showing that CLA captures all the essential features and reproduces all of the functionality of the Schema Mechanism. This application demonstrates that CLA's abilities are a superset of those of the Schema Mechanism. It also demonstrates that CLA is not only an accurate model of infant cognition, but can build sensorimotor schemas and thus is applicable to robot control. This work not only allows CLA to be used with mobile robots, but will for the first time allow the Schema Mechanism to be applied to a realistic robot and environment.

5.1 The Schema Mechanism

Drescher's Schema Mechanism is a constructivist learning system for situated agents that, like CLA, is based on Piaget's (1936, 1937) theory of infant cognitive development. Starting with

atomic sensory and motor primitives, the Schema Mechanism builds three different classes of constructs: schemas, synthetic items, and composite actions. These constructs are then used to create further constructs and thus, like CLA, the Schema Mechanism builds a hierarchical knowledge representation. Drescher used the Schema Mechanism to simulate an infant exploring its environment using vision, touch and taste. The Schema Mechanism has stood as one of the best implementations of constructivist learning, and the only known learning system to model constructivism as described by Piaget. This section describes how the Schema Mechanism works so that, in the next section, we can examine its weaknesses and see how CLA addresses those weaknesses.

5.1.1 Implementation Details

Prior to training, the Schema Mechanism starts with a set of primitive sensory *items* that reflect the state of the environment, such as `InFrontOfDoor` or `DoorOpen`. Each item is binary: it can be on or off. There is also a set of primitive *actions* whereby the agent can manipulate its environment, for example `OpenDoor`. As the system explores the environment by randomly selecting actions and monitoring the item states, the system can start to learn the association between item states and actions performed. These associations are represented as *schemas*. A schema is a prediction that given an initial set of item states (*context*), if a particular action is taken, then a certain change in item states can be expected (*result*). Drescher's schema is what Piaget would classify as a sensorimotor schema (section 2.1.1). A schema is written as *context*/*action*/*result*, where *context* is a set of items that are either on or off (indicated by a prefix of + or -, respectively), *action* is the name of the action, and *result* is a set of items indicating the change in the context state that occurs (again, prefixed with + or -). As an example:

```
+InFrontOfDoor/OpenDoor/+DoorOpen                    (5.1)
```

Schema 5.1 says that when I am in front of a door, and I try to open the door, the door will become open. A schema is successful when the activation of its action, while the context holds, effects the item transitions in its result. Statistics are maintained for each schema, including the schema's reliability, which is the rate of successful activation. A schema is deemed reliable when its reliability is above a predefined threshold.

Schemas are created using a technique called *marginal attribution*. When the system first starts, a schema is created for each action called a *bare schema*. Bare schemas have an empty context and an empty result, along with an extended context and extended result. The extended result maintains two data for every item: 1) a positive-transition correlation, which is the ratio of an item being turned On when the schema's action is performed, and 2) a negative-transition correlation, which is the same statistic for items turning off. The extended context also maintains two statistics for every item: 1) the probability that the schema will succeed if that item is on, and 2) the same probability if the item is off. When an the positive-transition or negative-transition correlation of an extended result item crosses a predefined threshold, a new schema is "spun off" of the bare schema with that item in the result. Similarly, when a context item is found to make the schema significantly more reliable, another schema is spun off, with those items added to the context of the spin-off schema.

Schemas can be used to create new *synthetic items*, which are newly created items representing hypothetical states that make a schema more reliable. For example, schema 5.1 may not

always work as described. If, say, the door is sometimes locked, then schema 5.1 will sometimes fail. This fact may be difficult to learn, particularly if the door's locked state cannot be directly sensed. When a schema is unreliable, it becomes "reified" as a synthetic item, which represents the state of the environment in which the schema is successful. Thus, schema 5.1 can be reified by a synthetic item:

```
[+InFrontOfDoor/OpenDoor/+DoorOpen]                (5.2)
```

We could call synthetic item 5.2 "DoorOpenable." It is on when the world is in a state where the schema will succeed. When a new synthetic item like this is created, it is tracked for correlations with all actions, just like primitive items, as part of marginal attribution. Thus, synthetic items can be incorporated into new schemas. The relationship between items and schemas, where each is used to build the other, is how Drescher implements constructivism: features, in the form of items, are combined using schemas and form new higher-level synthetic items.

Finally, schemas also support the creation of *composite actions*. When a new schema is created, and its result is unique among all other schemas, a new composite action is created, with the unique result as the "goal." The schemas are then chained backwards from the goal by 1) finding all schemas whose result matches the goal, 2) finding the next set of schemas whose results match the contexts of the first set of schemas, and so on. A composite action then has a set of contexts from which the goal can be reached, and the actions needed to get from a given context to the goal. For example, we could create a composite action with the goal DoorOpen:

```
<+DoorOpen>                                        (5.3)
```

When composite action 5.3 is selected, it selects a schema that, given the current state of all the items, would result in a item state change that would bring the agent one step closer to a state where the DoorOpen item is on. The action of that schema is then performed. If the goal is not achieved by the first step, then the composite action would again select an appropriate schema, bringing the agent another step towards the goal. If it is ever the case that there is no appropriate schema, or composite action takes too many steps and times out, then the action will fail. Baring these cases, eventually the door will be open.

To summarize, the Schema Mechanism starts with a set of primitive items and primitive actions. It then explores the environment to create a set of sensorimotor schemas. These schemas form the basis of new synthetic items. They also are used in the creation of goal-directed composite actions. Using these techniques, an agent can build a hierarchy of items to describe its environment, and a hierarchy of sensorimotor schemas that are combined into a plan for achieving some goal.

5.2 CLA and the Schema Mechanism

The Schema Mechanism is actually quite similar to CLA in spirit, if not in implementation. In this section, the two are compared as constructivist learning systems, the weaknesses of the schema mechanism are listed, and there is a discussion of how CLA addresses those weaknesses.

5.2.1 Similarities between CLA and the Schema Mechanism

Due to their common ancestry, CLA and the Schema Mechanism are analogous. In CLA, each layer consists of prototypes that represent a correlation of lower-level features in the environment. CLA's prototypes are analogous to Drescher's schemas that also use observed correlations to build a representation. The Schema Mechanism uses marginal attribution to build a complete schema one spin-off at a time. CLA uses the SOM to represent the features in the environment, where all presented features compete for representation in the SOM layer. Like the correlated knowledge represented in a schema's context, action and result, CLA represents correlated knowledge as the input weights for each node in a layer.

Just as the Schema Mechanism uses schemas as the basis for new hypothetical features (synthetic items) in the environment, each node in CLA becomes a feature that is presented to a higher-level layer via the activation vector. The Schema Mechanism creates new features from the knowledge of correlations found in the schemas. CLA' prototypes, which also hold correlative information, produce an activation vector that serves as an input feature to the next layer of SOMs. Like the feature generation of synthetic items, CLA takes correlation information and uses it to create new features that can be used in further learning.

CLA and the Schema Mechanism both learn correlations of features and use them to create new features. Integrating lower-level knowledge to form higher-level knowledge is the essence of constructivist learning, and captures the Information Processing Principles (Cohen, Chaput & Cashon 2002) described in Chapter 2. The methods used to perform these operations are different, however. Specifically, the method used by the Schema Mechanism is too inefficient to apply it to a realistic robot and environment The efficiency of the Schema Mechanism is discussed in the next section.

5.2.2 Efficiency of the Schema Mechanism

Where CLA and the Schema Mechanism differ is in the way they allow the knowledge base to grow. CLA clusters its knowledge in finite layers, representing different levels of representation, and learning takes place only one level at a time. The Schema Mechanism is not structured in this way; new schemas can be constructed at any level at any time, and there is no limit to the number of schemas per level, or the number of levels generated. As a consequence, the Schema Mechanism has four serious efficiency issues relating to the marginal attribution technique that restrict it to only simple models. These efficiency issues are not integral to constructivism in general, but apply only to the particular implementation used by the Schema Mechanism.

First, when a new schema is spun off from a source schema, all items must now be correlated with both the new schema *and* the source schema. Items can only be added to a schema's context or result one at a time. The new schema that results from adding an item then co-exists with its parent schema, meaning that a schema with n context items and result items will leave a trail, in the best case scenario, of $n+m$ intermediate schemas. Eventually, all combinations of contexts and results will be created, meaning that the growth is exponential. These intermediate schemas usually represent incomplete "stepping stone" correlations that do not occur in the environment, and their functionality usually subsumes that of their ancestors. Despite the vestigial nature of some intermediate schemas, they still figure into the statistical bookkeeping of marginal attribution. Intermediate schemas cannot simply be deleted, though, because they may represent a legitimate subset of correlations that do occur in the environment. The result is that the exponential growth of schemas makes the Schema Mechanism very resource intensive.

Compounding this problem, new synthetic items are added to the list of existing items which are being correlated with actions. The addition of new items to all schemas means that the introduction of a new item increases the number of correlations that must be computed by twice the number of schemas, one for contextual correlations and one for resulting correlations.

Moreover, each new composite action created produces yet another bare schema from which new schemas can be produced. All items are correlated with the performance of a composite action, and provide a new source for the exponential growth of schemas.

Finally, there is no process in the Schema Mechanism to prune items, schemas or actions. It may be possible to mark these constructs as inert and cease computations on them, or free them from memory altogether, but the method for pruning is not obvious, and Drescher does not propose one.

Because of marginal attribution, the Schema Mechanism will continually grow until its resources have been exhausted. Consequentially, the Schema Mechanism has been limited to simple domains like Drescher's "Microworld," a seven-by-seven grid that holds two objects that the agent can touch, see and taste. In order for the Schema Mechanism to be usable in realistic environments, a more efficient implementation is necessary. This efficiency is provided by CLA. CLA can not only implement the Schema Mechanism, but it improves upon it. The details of this implementation are described next.

5.2.3 Efficiency of CLA

CLA's use of SOMs makes generating knowledge more efficient. Rather than maintaining an ever-growing table of correlation data, the SOM provides a finite data space and a process for different representations to compete for that space. So the number of potential synthetic items can be constrained from the beginning.

CLA also has a staged learning system, where one level settles and allows the next level to learn. Once a level of SOMs has been trained, they can be harvested for candidate higher-level representations – schemas in this case. The lower-level SOMs can then be frozen and their resources can be used for the next level of SOMs.

Finally, CLA gives the modeler an alternative to the Schema Mechanism's everything-correlates-with-everything approach to schema building. CLA allows for specific groups of inputs, or "modes," to be sent to individual layers, further constraining growth of the knowledge base. For example, the inputs of the causal acquisition experiment in the previous chapter were separated into spatial and temporal modes.

Using CLA as the constructivist learning system makes the Schema Mechanism more efficient. Consequently, it becomes possible to apply it to more sophisticated domains. First, though, it must be shown that CLA can implement the Schema Mechanism and produce the same functionality reported by Drescher. The next section presents an implementation of the Schema Mechanism using CLA and compares it to the original Schema Mechanism using Drescher's agent and environment, the Microworld.

5.3 Implementing the Schema Mechanism with CLA

The Constructivist Learning Architecture Schema Mechanism, or CLASM, is an instance of CLA that implements the Schema Mechanism. CLASM uses the hierarchical SOMs of CLA to

implement the Schema Mechanism's marginal attribution process. The resulting implementation replicates all the functionality of the Schema Mechanism, demonstrating that CLA is not only an accurate cognitive model, but relevant to robot control. It also reimplements the Schema Mechanism more efficiently, allowing CLASM to be applied to realistic agents and environments, as will be demonstrated in the next chapter. This section includes the details of implementing the Schema Mechanism with CLA.

5.3.1 Implementation Details of CLASM

Like the Schema Mechanism, CLASM starts with a set of primitive items and primitive actions. While the Schema Mechanism starts with a bare schema for each action, CLASM associates each action with an *Action SOM*. The Action SOM acts as a mode for all contexts and results that hold before and after the performance of an action.

The weight vector for each Action SOM has twice the number of items that exists at the time of the SOM's creation. The first half of the weight vector represents the *context*, or the state of each item before the action is performed, represented by 1.0 (for on) or 0.0 (for off). The second half represents the *result*, or the change in item status that took place during the action's execution, which can range from –1.0 (an item has turned off) to +1.0 (an item has turned on). An Action SOM is only trained if the action associated with the SOM was just completed. Initially, an Action SOM is created for each primitive action. The initial input is the context and result of all primitive items (Figure 5.1).

Once the Action SOMs have been trained, they are harvested for schemas. When a node becomes a schema, the weight vectors are converted into lists of context and result items. Context items with weights less than 0.1 become negative context items, and those with weights greater than 0.9 become positive context items. Context items that are neither consistently on or off will receive weights between 0.1 and 0.9; these items are not used in the resulting schema. Result items with a weight change of greater than 0.9 become result items, positive if the change was positive (off to on), and negative of the change was negative (on to off). Result items that do not turn on or turn off consistently, and thus receive a weight value between 0.1 and 0.9, are not used in the resulting schema. A node becomes a schema only if there is at least one result item (Figure 5.2). If such a schema already exists, then the SOM node is represented by the existing schema. If the schema does not exist, it is created. Once harvested, Action SOMs are frozen.

If a schema has a novel result, a new composite action is created with that result as the goal. All reliable schemas are chained backwards from the schema with the novel result.

Once the schemas are harvested, a new stage of training begins. At the beginning of the following stage, new Action SOMs are created in association with every action, including the new composite actions. The input vector to the new Action SOMs is the context and result of every item, including the new synthetic items (Figure 5.3). A synthetic item is activated if its schema holds, i.e. if a) the schema's action is performed, b) all positive context items are true and all negative context items are false when the action is initiated, and c) all positive result items are on and all negative result items off when the action terminates.

In summary, CLASM is a implementation of the Schema Mechanism using CLA. CLASM replaces marginal attribution with CLA's layers of SOMs. This keeps the essential features of the Schema Mechanism while providing a more efficient implementation.

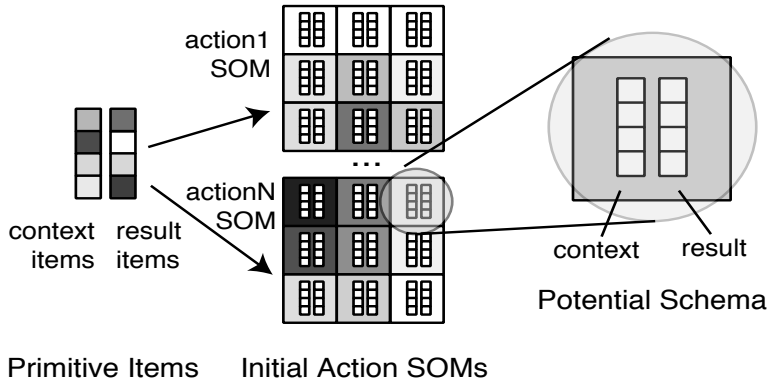

Figure 5.1: The initial state of CLASM. Clasm starts with an initial set of Action SOMs, one for each action, which are trained only when the corresponding action is performed. The input to each Action SOM is all the item states prior to the action (context) and the change in the item states after the action has completed (results). Each node in the Action SOM is a potential schema, each with a prototype context and result.

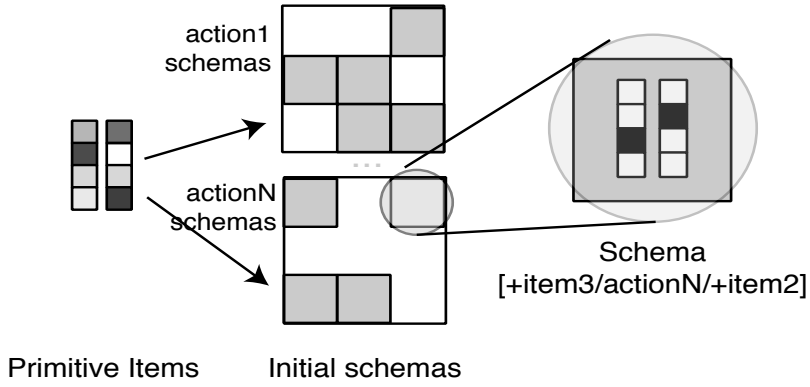

Figure 5.2: Action maps are harvested for schemas. After training, the Action Maps are harvested for schemas. Nodes that have at least one result item become candidate schemas. Items in the prototype context and result become context and result items in the schema if they pass a threshold. Duplicates are ignored. In the figure, the node that represents +item3/actionN/+item2 has become a schema.

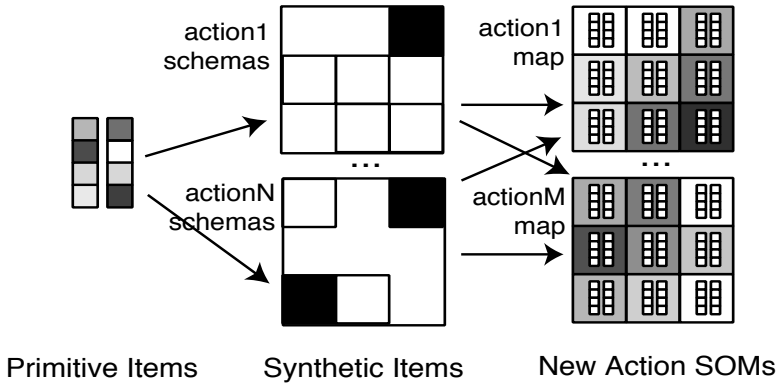

Figure 5.3: Training the second level in CLASM. Once the schemas have been harvested from the first level, the schemas are reified as synthetic items. A new set of Action SOMs is created, one for each action including synthetic actions. The input for these Action SOMs is the context and result of all items, including the newly created synthetic items.

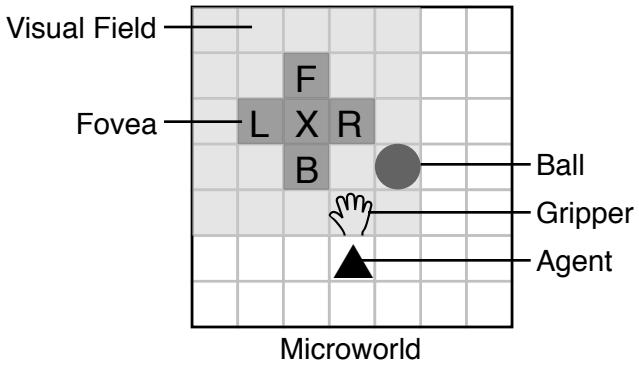

Figure 5.4: The Microworld. The Microworld is the simple agent and 7-by-7 environment used by Drescher (1991) and by the experiment in this chapter. Cell 0,0 is at the bottom left. The environment contains the agent, a gripper, and a ball. The visual field, in light gray, covers a 5-by-5 area and reports course visual information. At the center of the visual field is the fovea, in dark gray, which reports visual "details." The fovea has five sections: front (F), back (B), left (L), right (R) and center (X).

5.4 Experiment: CLASM in the Microworld

To demonstrate that CLA can build sensorimotor schemas from experience, CLASM is trained on the Microworld, a simple agent and environment used by Drescher (1991) to test the Schema Mechanism. Like Drescher's experiment with the Schema Mechanism, the purpose of this experiment is not have the agent learn some specific ability, but to learn a full and rich representation of the environment from fundamental features and actions. In this section, the Microworld is described, along with the details of the learning parameters used by CLASM. Finally the knowledge built by CLASM is reported and compared with the results from Drescher (1991).

5.4.1 Experiment Setup

Drescher (1991) implemented a simple agent and environment called the Microworld (Figure 5.4), which was used to test his Schema Mechanism. To ensure that the CLASM is producing the same results as the original Schema Mechanism, it will be tested in the same environment. The Microworld, first described by Drescher (1991) is described here again.

The Microworld is a two-dimensional, seven-by-seven grid, with cell 0,0 at the bottom left. All objects in the Microworld are one grid cell in size. There are three objects in the Microworld: a ball, an agent, and the agent's hand. The ball is stationary, but can be picked up and moved by the agent (via its hand).

The agent is immobile, but has a hand, along with visual, tactile, taste and proprioceptive sensory systems. The vision system provides the agent with visual information about the grid within the vicinity of the agent. First, the vision system reports the presence of objects within a five-by-five visual field. Second, this visual field has a "fovea" which reports "details" — features not seen outside the fovea — of objects at the center of the visual field and the four adjacent field grid cells: left, right, forward and back. Drescher used 16 arbitrary details, but there are only three objects in the world, so only 3 details are used in this experiment, one for each object. The center of vision can be moved around the Microworld only one cell at a time and only in the four cardinal directions. The agent can sense where the visual field is centered via proprioception.

Primitive Action	Function
handf, handb, handr, handl	These actions move the hand forward, backward, right and left, respectively.
eyef, eyeb, eyer, eyel	These actions move the eye forward, backward, right and left, respectively.
grasp	This action closes the hand. If the hand is in the same cell as an object, and the hand wasn't already closed, then the hand will grasp the object. A grasped object moves with the hand.
ungrasp	This action opens the hand, releasing any object it might hold.

Table 5.1: Primitive actions in the Microworld. This table lists the ten primitive actions available to the agent in the Microworld. These actions allow the agent to control its hand and gaze.

The hand can also be moved one cell at a time in the four cardinal directions. The hand is limited to a three-by-three area directly in front of the agent. (The hand cannot be beside or behind the agent.) The agent can sense the position of the hand via proprioceptive feedback, and can also see the hand if it happens to be looking in its direction. The hand can also report tactile information of objects that are directly adjacent to the hand. The hand can be opened and closed, and the agent can sense whether the hand is opened or closed. When the hand occupies the same grid cell as the object and is closed, the object is grasped by the hand such that, when the hand moves, the object moves with it. The hand can then be opened, depositing the object. The hand can report on tactile "details," similar to visual details, that are features of objects that are only available when of an object is in the same grid cell as the hand (regardless of whether the object is held or released). There are four tactile details, 0 through 3, and the ball has details 1 and 3. (Tactile details 0 and 2 are unused.)

Finally, the agent itself can feel objects that are directly adjacent to the body of the agent. Objects that are directly in front of the agent and grasped can also be tasted.

The agent is given a set of primitive actions, described in Table 5.1, which allow it to move its eye, its hand, and to open and close its hand. The agent senses the world through a set of primitive items, listed in Table 5.2. These items allow the agent to sense the environment via vision, touch, taste and proprioception.

At the start of each simulation, the agent is positioned at cell 3,1. The hand and the ball were started in random positions within the hand's reach. The hand and the ball were never started at the same place.

5.4.2 Learning Parameters

CLASM alternated between sensing the world and selecting an action. Actions were selected at random. Each action map was 10-by-10 nodes in size. The learning parameters of each map followed the standard SOM training procedure of gradually reducing the learning rate and neighborhood until there was little or no change in the layer, which usually took about 10,000 iterations. All layers at each level had identical learning parameters.

Once the first level had been trained, the training on layers in the first level was stopped, and the new action maps were created. These maps followed the same training regimen. Two levels were trained.

Item	Meaning
hp11,..., hp33	Haptic-proprioceptive (hand-position) items, one for each possible hand position. Position (1,1) is the lower left corner of the range.
vp11,..., vp33	Visio-proprioceptive (eye-position) items, one for each possible glance orientation. Coordinate designates center of visual field, using the same conventions as for hand position.
tactf, tactb, tactr, tactl	Coarse tactile items, one for each side of the hand (front, back, right and left).
text0, text1, text2, text3	Detailed tactile items, denoting arbitrary textural details on an object in the same cell as the hand.
bodyf, bodyb, bodyr, bodyl	Coarse tactile items, one for each side of the body (front, back, right and left).
taste0, taste1, taste2, taste3	Taste items, designating arbitrary surface details of an object touching the mouth (front edge of the body).
hcl	Hand closed.
hgr	Hand closed and grasping something.
vf00,..., vf44	Coarse visual-field items, one for each of 25 cells. Region 0,0 is at the lower left.
fovf0, fovf1, fovf2 *fovb0, fovb1, fovb2* *fovr0, fovr1, fovr2* *fovl0, fovl1, fovl2* *fovx0, fovx1, fovx2*	Visual details corresponding to each of five foveal regions: front, back, right, left and center. Each has three details.

Table 5.2 Primitive items in the Microworld. This table lists the set of primitive items that are available to the agent in the Gridworld. These items allow the agent to see, touch, taste, and know the location of its own hand and eye relative to the agent (proprioception).

5.4.3 Results

The training session resulted in a hierarchy of knowledge that reproduced all of the functionality reported by Drescher. This demonstrates that CLASM is indeed a reimplementation of the Schema Mechanism. CLASM also produced many knowledge structures that were *not* reported by Drescher, but this does not necessarily mean that they were not generated by the Schema Mechanism: they may have been generated and not reported. However, the output that Drescher chose to highlight represents what he considered the essential features of the Schema Mechanism. CLASM has these same essential features, which are described in detail below.

Following is a summary of the knowledge generated by CLASM, along with an interpretation of their meaning. Refer to tables 5.1 and 5.2 for the meaning of individual items and actions. Also included is the page number in Drescher (1991), where these results were originally reported. While many of the schemas generated by CLASM are identical to those reported by Drescher, some of the schemas reported below are variants but which still demonstrate the fundamental capability that Drescher illustrates with his results. All of the capabilities reported by Drescher are also produced by CLASM.

Grasping (p120)

These schemas are examples of the simplest schemas:

```
/grasp/+hcl,                                    (5.4)
+text0+text2/grasp/+hcl+hgr.                    (5.5)
```

Schema 5.4 says that when the agent closes its hand (grasp), the agent's hand is closed (+hcl). There are no context items for schema 5.4 because this schema always works. Schema 5.5 says that when the agent can feel something (+text0+text2, the tactile details of the ball), and it closes its hand (grasp), the agent's hand is closed and grasping something. In this case, grasping something (+hgr) is dependent on feeling something (+text0+text2).

Elaborating the visual field (p122)

These schemas report the change in relative position of seen objects when the eye moves:

```
+vf21/eyer/+vf11-vf21                           (5.6)
+vf30/eyer/+vf20-vf30                           (5.7)
```

Schema 5.6 says that, given an object at 2,1 (+vf21), if the agent's eye moves to the right, then the agent will see an object at 1,1 (+vf11) and it will no longer see an object at 2,1 (-vf21). Schema 5.7 works in a similar way. CLASM is learning the relationship between what is seen and moving the eye. These relationships can be grouped together to fully describe the position of an object throughout all four eye movements:

```
+vf34/eyer/-vf34+vf44,                          (5.8)
+vf44/eyel/-vf44+vf34,                          (5.9)
+vf44/eyef/+vf43-vf44,                         (5.10)
+vf43/eyeb/+vf44-vf43.                         (5.11)
```

By mapping out the visual field in this way, CLASM can build a plan for moving the eye that will put the object in an arbitrary position in the visual field.

Foveal relations (p124)

CLASM also learns the same sort of visual relationship between the areas in the fovea:

```
+vp23+vf22+fovb0/eyeb/+vp22-vp23-fovb0+fovx0.    (5.12)
```

Schema 5.12 says, given that the agent sees something directly behind the center of vision (+fovb0), and the eye is moved back (eyeb), then the agent will be looking directly at the object (fovx0). All foveal relations of this nature are generated.

Elaborating the proprioceptive fields (p126)

CLASM also learns the relationships between moving they eye and sensing where the eye is looking. The following is a network of visual schemas similar to 5.8 through 5.11, except that rather than sensing an object, the eye is sensing itself:

```
+vp23/eyer/-vp23+vp33,                    (5.13)
+vp33/eyel/-vp33+vp23,                    (5.14)
+vp33/eyeb/-vp33+vp32,                    (5.15)
+vp32/eyef/+vp33-vp32.                    (5.16)
```

Schema 5.13 says that, given that the agent's eye is at 2,3 (+vp23), if the eye is moved to the right (eyer), then the agent's eye is at 3,3 (+vp33) and no longer at 2,3 (-vp23). Schemas 5.13 through 5,16 and others like them would allow CLASM to point the eye at any coordinates given an arbitrary starting location.

A similar set of schemas is produced for hand movements and hand proprioception:

```
+hp23/handr/-hp23+hp33,                   (5.17)
+hp33/handl/+hp23-hp33,                   (5.18)
+hp22/handf/-hp22+hp23,                   (5.19)
+hp33/handb/-hp33+hp32.                   (5.20)
```

Schema 5.17 says that, given the agent's hand is at 2,3 (+hp23), if the agent moves its hand to the right (handr), its hand will be at 3,3 (+hp33) and no longer at 2,3 (-hp23). These relations exist for every movement action between all positions, allowing for arbitrary hand placement.

Negative consequences (p127)

In this case, an action will turn an item off. We have already seen many examples above. Here are three more:

```
+vp31+vf10/eyel/+vp21-vp31-vf10,          (5.21)
+vp21/eyel/+vp11-vp21,                    (5.22)
+hp13+vf24/handb/+hp12-hp13.              (5.23)
```

Schema 5.21 says that, if the agent is looking at position 3,1 (+vp31) and it sees something at 1,0 (+vf10), and the eye moves to the left (eyel), then the agent is now looking at 2,1 (+vp21) and not at 3,1 anymore (-vp31), and there is no longer any object at 1,0 (-vf10). The object just disappears because it has exited the visual field.

Schemas 5.22 and 5.23 are variants of the visio-proprioceptive and haptic-proprioceptive schemas above, respectively.

Positional actions (p127-129)

Once the first stage is complete, the results of these schemas form the goals of new composite actions. CLASM produces the same actions reported by Drescher:

```
New action: <+hp22>,                      (5.24)
New action: <+hp33>,                      (5.25)
New action: <+vp22>,                      (5.26)
New action: <+vp33>,                      (5.27)
New action: <+vf34>,                      (5.28)
New action: <+vf12>.                      (5.29)
```

Each of these composite actions is described by a goal state. In the case of 5.24, the goal this composite action is to have the hand be at position 2,2. Within this and every composite action is a network of schemas that can build a plan for getting from the current state to the goal state.

Moving the hand to the mouth (p129)

CLASM has also learned how to pick up objects and move them to the mouth:

```
+hp22+vp22+text1+text3+hcl+hgr
+vf22+fovx1+fovx2/handb/
+hp21-hp22+bodyf+taste1+taste3+vf21
-vf22+fovb1+fovb2-fovx1-fovx2.                    (5.30)
```

Schema 5.30 is the final step in moving an object to the mouth. The context of schema 5.30 say that the hand is at position 2,2 (+hp22), they eye is also at 2,2 (+vp22), the hand can feel the ball (+text1+text3), the hand is closed and grasping something (+hcl+hgr), there is something directly at the center of the visual field (+vf22), and specifically there is a hand and the ball at the center of the fovea (+fovx1+fovx2). Given this context, if the agent moves the hand back, then the hand will now be at 2,1 (+hp21-hp22), the ball will be against the front of the agent's body (+bodyf), it can taste the ball (+taste1+taste3), the object has moved in the visual field from 2,2 to 2,1 (+vf21-vf22), and the ball and hand have moved from the center of the fovea to the back of the fovea (+fovb1+fovb2-fovx1-fovx2).

The importance of schema 5.30 is that now +taste1 can become the goal of a composite object. This will allow CLASM to start from a large number of initial item states and build a plan to put the ball in the agent's mouth.

Visual effects of incremental hand motions (p130-131)

CLASM has learned how the visual field is altered when the hand moves:

```
+vf11/handl/+vf01-vf11.                           (5.31)
```

Schema 5.31 says that, if the agent's hand is at 1,1 (+vf11) and it moves its hand to the left (handl), then the agent will see something at 0,1 (+vf01-vf11). Notice that the agent doesn't know what it is seeing, either before or after the hand moves. This is because cell 0,1 of the visual field is outside the fovea, so the details of the object cannot be discerned. Regardless, CLASM has learned this regularity. If the hand moves within the foveal region, then the schema becomes more detailed:

```
+hp22+vp32+vf11+fovl2/handr/
-hp22+hp32-vf11+vf21-fovl2+fovx2.                  (5.32)
```

Schema 5.32 says that, given the agent's hand is at 2,2 (+hp22) and the eye is at 3,2 (+vp32) and it sees something at 1,1 (+vf11) that turns out to be the hand on the left side of the fovea (+fovl2), then if I move the hand to the right (handr), now the hand is at 3,2 (-hp22+hp32), the

object has moved from 1,1 to 2,1 (-vf11+vf21) and that object is the hand moving from the left of the fovea to the center (-fovl2+fovx2).

Touching what is seen (p132)

CLASM also learns to touch what it sees with the help of a composite action. A schema using the composite action <+fovx1> can be used to move the hand to touch an item just below the center of vision:

```
/<+fovx1>/+tactb+fovx1.                    (5.33)
```

Schema 5.33 says that, if the agent moves the hand to the center of its vision, it will feel something just below the hand. This schema only succeeds when there is something just below the center of vision. It may fail (say, if the agent tries to affect <+fovx1> by moving the eye and not the hand), but its success will reveal the position of the ball and how to touch it.

Similar schemas can move the eye to see what is touched.

Palpable and visible persistent objects (p136-137)

These are synthetic items that represent the beginning of a persistent-object concept for the agent.

```
[/<+hp21>/+hp21+tactl].                    (5.34)
```

In synthetic item 5.34, the composite action <+hp21> is used to create the hypothetical feature "Palpable object at 1,1." In other words, when 5.34 is true, then the supporting schema is predicted to work. If that schema works, then that means the agent can be confident that, were it to move its hand to 2,1 (<+hp21>), its hand would end up at 2,1 (+hp21) and it would feel something to its left (+tactl). Thus, turning on synthetic item 5.34 means that there is an object at 1,1. Note that the agent does not necessarily have to move its hand to determine this fact about the world. Synthetic item 5.34 represents the a higher-level feature in the world that would make the underlying schema true.

Similarly, the following synthetic item represents "Visible object at (3,3)":

```
[/<+vp22>/+vp22+vf33].                     (5.35)
```

These, and others, combine to form groups of synthetic items that correspond to an object's persistent identity.

Cross-modal representations (p140)

Finally, these schemas represent persistent-object information across modalities:

```
+vp12+fovb1/<+hp21>/+hp21+tactl,           (5.35)
+hp21+tactl/<+vp12>/+vp12+fovb1.           (5.36)
```

Schema 5.35 says that, given the agent's eye is at 1,2 (+vp12), and the agent sees the ball at the back of the fovea (+fovb1), then if the agent moves its hand to 2,1 (<+hp21>), the hand will end up at 2,1 (+hp21) and it will feel something to the left of the hand (+tactl). Schema 5.35 connects the visual information in the context with the tactile information in the result by moving the hand. Conversely, schema 5.36 connects the tactile information in the context with the visual information in the result by moving they eye.

5.4.4 Experiment Discussion

CLASM succeeded in building a rich and full model of the agent and its environment, reproducing all of the reported capabilities of the original Schema Mechanism. This result demonstrates both that CLA can learn sensorimotor schemas and that CLA is a reimplementation of the Schema Mechanism. CLA reproduced the capabilities of the Schema Mechanism while imposing hard restrictions on the growth of the knowledge base to well-defined layers of schemas. What makes CLA more efficient than the Schema Mechanism is how it bypasses simple schemas with one or two result items, and jumps directly to more complex, multi-result and multi-context schemas. There is no need to go through the stepwise building process to get to more complex schemas because the entire context and result are being presented to the SOM at once. The resulting complex schemas simply emerge from the data. Simpler schemas are easy to derive from the final product, should they be desired.

It is worth noting that the original Schema Mechanism pushed the limits of a Thinking Machines CM2 computer with 2 gigabytes of memory and 65,536 physical processors operating in parallel, while all the experiments in this dissertation were performed comfortably on a Apple PowerBook with a single G4 processor and 640 megabytes of memory. This is not enough information to draw a scientific conclusion, and the difference in architectural and programming styles does not support a direct comparison. However, it's reasonable to speculate that CLASM puts less demand on a given platform than the Schema Mechanism, and that these differences stem in part from their different approaches to implementing constructivism.

5.5 Conclusion

CLA has been shown to build both propositional and sensorimotor schemas, making it relevant to agent control. In addition to being an accurate model of infant cognitive development, CLA can build a grounded, rich and full model of a simple agent in a simple environment through experience. The implementation used to do build this model is a reimplementation of the Schema Mechanism. This reimplementation is both faithful to the original and more efficient. CLA's ability to build a grounded model of the environment with sensorimotor schemas will allow it to be applied to a realistic robot in the next chapter. CLA's adherence to the Information Processing Principles (Cohen, Chaput & Cashon 2002) will also allow a robot to have the adaptability and fallback that are necessary components of robust control. This is demonstrated in the next chapter.

6. Learning with Delayed Rewards

In Chapter 4, CLA was used to model infants' acquisition of causal perception, demonstrating that CLA learned hierarchical knowledge, had stage-like development, and supported graceful degradation using fallback. In Chapter 5, CLA was used to control a simple agent, demonstrating that CLA can also learn sensorimotor schemas making it applicable to robot control. In this chapter, these two ideas come together: CLA is used to control a robot that learns how to forage by building a hierarchical representation of its environment, and responds to unexpected damage with graceful degradation.

So far, CLA has been used as an unsupervised, hierarchical learning system for propositional and sensorimotor schemas. In this chapter, CLA will learn goal-directed behavior with delayed rewards. CLA is a new kind of skill learning system for such domains. In addition to assigning value to schemas, like any Reinforcement Learning system, CLA generates new synthetic items that are used to make these schemas more reliable. This chapter presents two experiments that demonstrate this novel approach to learning with delayed rewards, and show that such a system provides a robot with robust control using fallback.

The first section discusses the challenges specific to learning with delayed rewards, and compares CLA's approach to other standard Reinforcement Learning approaches. The next section describes an experiment where a Pioneer robot, simulated on an externally verified platform, is trained to forage for specimens. The knowledge that is learned, and how it applies to the goal of foraging, is discussed in detail. The third section describes another experiment in which the trained robot is damaged, and the performance of the robot is compared to pre-damage levels. This is similar to the fallback experiment in Chapter 4, and demonstrates that CLA is a robust robot controller. Finally, CLA is compared to other related work.

6.1 CLA and Delayed Rewards

The SOM, the central component of CLA, is an unsupervised, self-organizing learning system. CLA does not make use of normative feedback or an error signal to build knowledge. This is important for a large tract of cognitive modeling and machine learning, where environmental feedback is unobtainable or nonexistent. Up to this point in this dissertation CLA has been used as a system that learns a hierarchical representation through observation without feedback.

However, if the robot is to perform some goal-directed behavior, unsupervised learning is not enough. Unsupervised learning is a value-free approach to machine learning, but goal-directed behavior implies that at least one thing — the goal — has a value. An unsupervised learning system, like the unmodified SOM, can give a robot choices, but it cannot help the robot decide which choice to take. In this section, a system is described for choosing between schemas, and this system is compared to standard Reinforcement Learning techniques.

6.1.1 Constructivist Reinforcement Learning

To support goal-based learning, CLA uses a technique for assigning value to items originally suggested by Drescher (1991), but never implemented or tested. CLA allows primitive items to be assigned an *inherent* value, indicating how desirable it is for that item to be true. Using the CLASM implementation described in the previous chapter, CLA discovers schemas that have valuable items in their results. These schemas are then assigned the value of their result. Because we know, with some degree of reliability, that the context of a valuable schema can lead to the activation of valuable items, the context items can receive a *deferred* value. Thus, schemas can be chained backwards by matching the context of a valuable schema with the result of another schema. Any items which are connected to the valuable result are assigned a deferred value. In addition, the reliability of each schema is maintained. To select an action, the most valuable of the applicable schemas is chosen and, in the event of a tie, the most reliable of those is chosen.

This approach is similar to the ideas at work in Reinforcement Learning algorithms such as Q-Learning (Watkins & Dayan 1992; Sutton & Barto 1998). Q-Learning determines the value of state/action pairs through exploration. Values are propagated throughout the state space using a temporal differences algorithm.

Unlike Q-Learning, however, CLA builds entirely new high-level features that represent the reliability of a schema. A schema's context and action are essentially a state/action pair that receives its value from the result. Higher-level features — or synthetic items as they are called by Drescher — are hypothetical states in the environment that indicate the reliability of their source schemas. Thus, if a synthetic item indicates the reliability of a valuable schema, then the synthetic item is also valuable. These new items can be included in the contexts of newer, higher-level schemas, and can receive their own deferred value.

Creating new features is something that standard Reinforcement Learning algorithms simply do not address. High-level features are not chosen by the programmer but discovered by the learning algorithm. CLA elaborates on the original state space by discovering hidden features that describe the applicability of its schemas. These new features can then be used in a new set of higher-level schemas, which become the source of still more higher-level features. Without an adequate set of states, even the simplest Reinforcement Learning problem can be unsolvable. Elaborating the state space can give Reinforcement Learning the features needed to learn an important policy. Additionally, by extending learning with delayed rewards to higher-level constructed representations, CLA can build more sophisticated control based on unseen states, and give the robot a strategy for handling noisy input.

This is not to say the CLA will learn a policy faster or more efficiently than Q-Learning or some other Reinforcement Learning algorithm, and no comparison based on speed or efficiency is implied. Instead, CLA offers a capability that is orthogonal and complementary to Q-Learning. Indeed, the approach to learning from delayed rewards presented here could probably be replaced with a more sophisticated Reinforcement Learning algorithm to improve the system even further.

But that is not a proposed thesis and thus is beyond the scope of this dissertation. (However, it does suggest an avenue of future research; see section 7.1.5.)

6.1.2 Robust Reinforcement Learning

Using CLA for learning with delayed rewards also brings the advantages of fallback and recovery. A Q-Learning system will develop a policy based on a single set of sensors and actions. But if the underlying tenets of the policy are compromised by a change in the environment, then all of the policy may fail at once, without any behavior to fall back on other than random exploration.

CLA, on the other hand, builds a policy in layers from most general to most specific. As shown in section 4.4, when the nature of the input changes, the more specific higher levels may fail, but the lower-level knowledge still remains somewhat applicable. Fallback gives a robot a policy for action — even if it is a less efficient one — when the most specialized knowledge becomes inapplicable. Not only does fallback provide a method for graceful degradation, but it also facilitates recovery. While algorithms like Q-Learning will need to resort to random exploration to rebuild its knowledge base, CLA can reassess the reliability of its schemas and start making better choices right away. This decreases the relearning time, and the time spent by the robot in suboptimal behavior.

6.1.3 Summary

CLA is a new approach to learning from delayed rewards. It brings hierarchical learning to standard Reinforcement Learning algorithms, allowing for policies to be built on hypothetical features in the environment. It also brings fallback to these learning systems, making the robot more resilient to changes in the stimuli, and expediting the recovery process.

6.2 Experiment: Foraging

To demonstrate learning with delayed reward, a Pioneer robot uses CLA to learn how to foraging for specimens. The robot is equipped with a camera and grippers, and it must locate, approach and collect specimens. Just as with the Microworld in the previous chapter, CLA is expected to learn a set of lower-level schemas, and use those schemas to build a set of higher-level schemas. In addition, CLA will learn which schemas will reliably result in a valuable outcome. This experiment will demonstrate CLA's ability to learn with delayed rewards.

6.2.1 Robot and Environment

The robot in this experiment is a Pioneer 2-DX robot that is simulated by Stage (Vaughn 2000), an externally validated robot simulator (figure 6.1). The robot is equipped with a differential drive, active grippers, bump sensors, and a camera using the CMVision blob tracker (Bruce et al 2000) to detect blobs of color. The grippers are equipped with a laser to detect the presence of an object between the paddles of the gripper. The robot is placed in a circular room with 28 specimens, which are identifiable by their color. CLA communicates with the Stage simulator using the Player robot device server (Gerkey et al 2000; Gerkey et al 2003). The robot's camera has a 60°

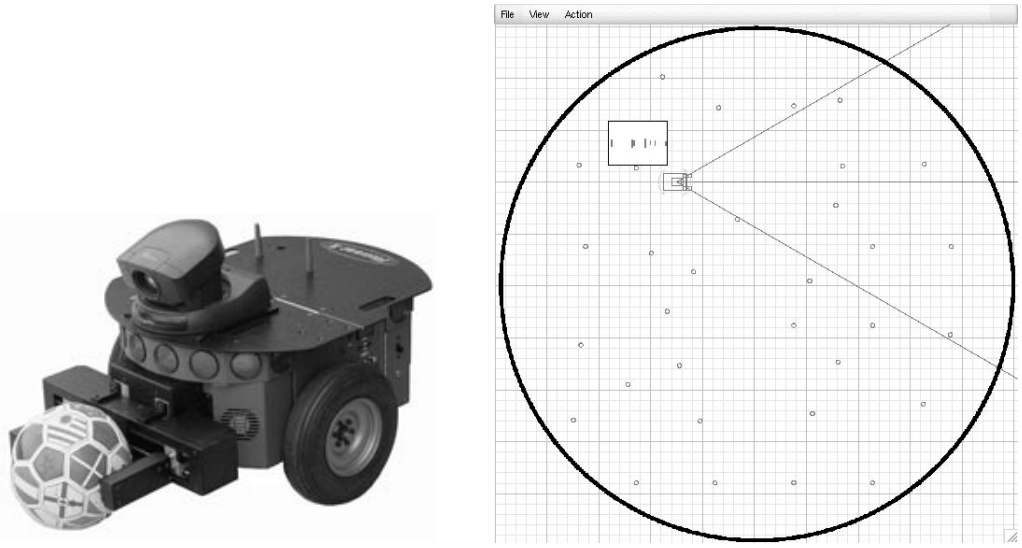

Figure 6.1: The Pioneer 2-DX, real and simulated. The robot used in this experiment is a Pioneer 2-DX with a differential drive, a camera, and a pair of active grippers. The actual robot is picture on the left. The robot is simulated using Stage, an externally validated robot simulator. On the right is the display window from the Stage simulator. The robot is the rectangular box. The camera has a 60° viewing angle, as diagramed by the diagonal lines coming from the robot. The Blob View, pictured above it and to the left, shows the available "blobs" in the CMVision blob tracker.

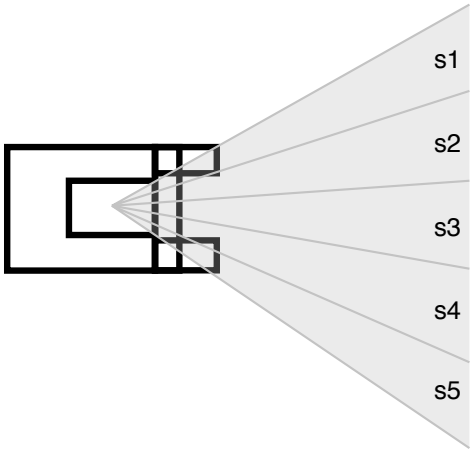

Figure 6.2: The robot's visual system. The robot has a 60° viewing angle, shown in gray above. The image is separated into five sectors of 12° each. When a robot detects a blob in one of these sectors, then the corresponding item (s1 through s5) becomes true.

viewing angle which is divided into 5 sectors of 12° each (Figure 6.2). The blob tracker can detect the presence of a specimen in each of these five sectors.

The robot has six primitive actions (Table 6.1) that allow the robot to move forward or backward by 50cm (*forward* and *backward*), turn left or right 5° (*turnl* and *turnr*), turn completely around (*turnaround*), and close its grippers (*grip*). The robot also has seven primitive items (Table 6.2) for observing its environment. Five of these items, *s1* through *s5*, indicate that a specimen is within one of the cameras sectors, ranging from far left to far right. The other two items allow the

Primitive Action	Function
forward, backward	Moves the robot forward or backward 10cm.
turnl, turnr	Turns the robot 5° to the left or to the right.
turnaround	Turns the robot 180°.
grip	Closes the grippers and reopens them. If a specimen is within the grippers, it will be obtained.

Table 6.1: Primitive actions in the foraging experiment. This table lists the six primitive actions available to the agent in foraging experiment. These actions allow the agent to move around the space and acquire specimens.

Item	Meaning
s1, s2, s3, s4, s5	Triggered by the blob tracker, these items indicate that a specimen is visible in the camera. The different items specify where the specimen is in the field of view: on the far left (s1), on the near left (s2), in the center (s3), on the near right (s4) or on the far right (s5).
ingrip	Triggered by the gripper laser beam being broken, this indicates than an object is within the grippers.
bump	Triggered by the bump sensors surrounding the robots body, this indicates that the robot has struck a wall.

Table 6.2 Primitive items in the foraging experiment. This table lists the set of primitive items that are available to the agent in the foraging experiment. These items allow the agent to see specimens, sense when they are in the gripper, and sense when the robot has struck a wall.

robot to sense when the gripper's beam bas been broken (*ingrip*) and when the bump sensors are triggered (*bump*).

The robot has two built-in reactive behaviors. First, whenever the robot detects an object in its grippers (*ingrip*), it automatically closes the grippers and acquires the specimen (*grip*). Second, whenever the robot runs into a wall (*bump*), it backs up and turns around (*back* and *turnaround*).

6.2.2 Learning System

This experiment uses the CLA implementation of the Schema Mechanism (CLASM) described in the previous chapter, plus the value distribution mechanism described above (section 6.1). One item, *ingrip*, has an inherent value of 10.0. For any schema that results in some valuable item (such as ingrip) being 'on', the context items of that schema received a deferred value that is 95% of the resulting value. Values are propagated through all available schemas. An item that receives multiple values (by being in various places in the schema chain) retains the maximum of the values. Deferred values are generated whenever a new schema is created.

In addition, the method of building sensorimotor schemas was altered. CLASM used the Schema Mechanism's method of including an item in the result only if that item changed (either from on to off or vice versa) as a result of the schema. This worked fine in the Microworld, where all locations and positions are discrete and movement necessarily causes a change. But the environment used in this experiment is continuous, and the resolution of the sensors low, so movement in this environment might not result in a change. Consequently, using the changed-item approach to schema building would keep many important schemas from being learned. For this experiment, CLA trained simply on the item states for both the context and the result.

6.2.3 Action Policy

An optimal action can be selected by choosing the most valuable applicable schema, that is, the schema with the most valuable result whose context matches with the current environment state. In the event of a tie, the schema with the highest reliability is chosen. If there is no applicable schema, an action is chosen at random.

During training, it is useful to find schemas that are known to work and explore them more fully by using them more often. However, if the system always chooses the most valuable schema, it won't have an opportunity to discover potentially more direct and more valuable strategies. Adding a small amount of exploration, say 30% explore vs. 70% goal-directed, will allow CLA to eventually learn a full model of the environment. However, a greater amount of exploration speeds up the learning process. Thus, during training CLA uses the most valuable schema 20% of the time, and selects a random action the other 80% of the time.

6.2.4 Training Parameters

Like CLASM in the previous chapter, the system is trained on context-result pairs for each action. Each action has its own SOM that are each 20x20 nodes in size. The system was trained until there was minimal change in the first level of representations, usually for 10000 actions. After 10000 training epochs, schemas were harvested from the first level, synthetic items were created for each schema, and a second level of Action SOMs was created and trained. For each layer, the neighborhood was steadily decreased from 15 to 0, while the learning rate was simultaneously decreased from 0.3 to 0.01.

To test CLA's performance, the knowledge base was stored every 1000 training epochs for 10 different training runs. This knowledge base was loaded into a robot that was placed in the training environment and run for 3000 actions. The number of captured specimens was counted. This statistic was measured three times for each of the 10 knowledge base snapshots.

6.2.5 Experiment Results

CLA quickly built up a basic set of schemas required to target and acquire a specimen. Not surprisingly for such a simple task, CLA developed a policy for maximizing value after only a few thousand steps. CLA also built higher-level items that further refined the state space. These items were used to build higher-level schemas that were more efficient and more reliable than the lower-level schemas. The details of the learned knowledge, and the reliability of the schemas, are described below.

CLA's performance is graphed in Figure 6.3. CLA learned its best lower-level policy after about 3000 training epochs. Upper-level schemas started training after 10000 training epochs. The upper-level schemas significantly improved performance ($M = 22.9$, $S.D. = 1.19$) over lower-level schemas by themselves ($M = 20.4$, $S.D. = 1.53$), $t(18) = -2.55$, $P = 0.02$ (two-tailed).

6.2.6 Discussion

CLA developed a set of schemas that gave the robot useful abilities and enhanced its performance. This section discusses these results by first highlighting some of the robot's learned abilities, including navigation, hidden features, feature refinement, and object persistence. The robot's performance is then discussed in relation to these abilities, and the contribution of lower-level schemas and upper-level schemas are compared. All of the lower level schemas described

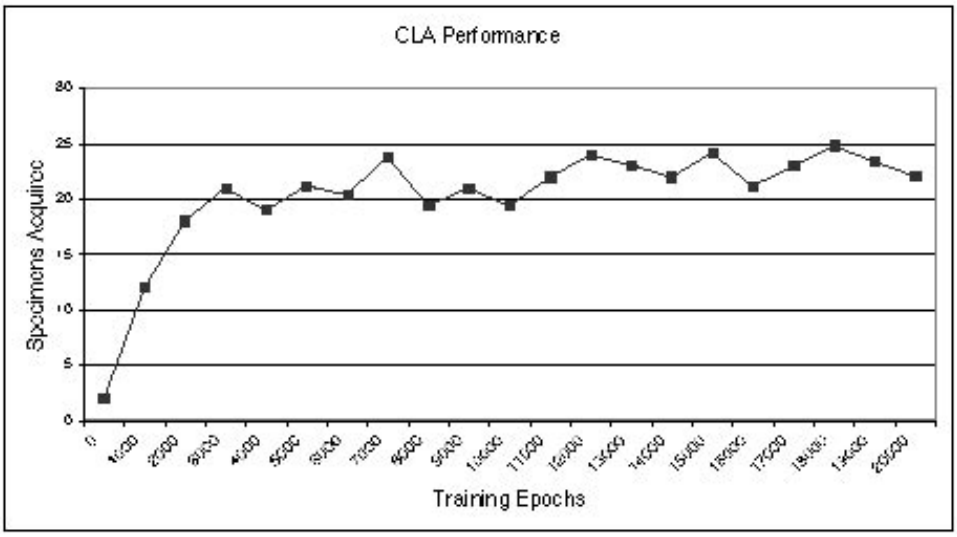

Figure 6.3: Performance at different stages in learning. The graph shows the number of specimens acquired during 3000 actions using the schemas available at different points in training. Level 2 started training at 10000 epochs. CLA quickly learned an effective strategy for acquiring specimens using lower-level schemas. Once Level 2 started training, CLA then learned a set of higher-level schemas that increased the robot's performance.

below were generated in every one of the 10 experiments. Some of the higher level schemas did not develop in some of the runs, but six runs generated all of the higher-level constructs reported below. Because the items s1 through s5 are mutually exclusive, some schemas in this section are abbreviated for readability by removing redundant negative context and result items.

Robot Navigation

As the robot explored the space, CLA quickly built up simple turning schemas that would move a specimen from one part of the visual field to the other. For example:

```
+s1/turnl/+s2.                                    (6.1)
```

Schema 6.1 states simply that when there is a specimen in Sector 1, and the robot turns left, then there will be a specimen in Sector 2. Schemas like these were learned for all sectors and both turn commands. These schemas were not very reliable (M = .286, S.D. = .012) for reasons that are discussed later.

The robot also learned that moving forward while the specimen is in front will leave the specimen in front:

```
+s3/forward/+s3.                                  (6.2)
```

Schema 6.2 is a strategy for approaching a specimen that is centered in the camera, and is the most used schema for acquiring a specimen, although it too is not very reliable (M = .351, S.D. = .012). Schemas like 6.1 and 6.2 were built very early during training. However, actually acquiring a specimen is much more rare, and took longer to learn. Since ingrip is the only item with any value, and no schema had yet been constructed that resulted in ingrip, these schemas had no value

Item	Deferred Value
s3	9.5
s2, s4	9.025
s1, s5	8.574

Table 6.3 Deferred value of primitive items after training. This table lists the deferred value after training of the primitive items in the foraging experiment. Using CLA's system for learning with delayed reward, each item eventually received a value determined by their distance from a valuable item (ingrip) through the available schemas. These values were assigned to these items consistently (S.D. = 0) for every run.

associated with them. Once a strategy to activate ingrip had been learned, however, CLA had its first valuable schema:

```
+s3/forward/+s3+ingrip.                                    (6.3)
```

With the creation of Schema 6.3, the value of ingrip was propagated back through all the schemas (section 6.1), including schemas 6.1 and 6.2. The post-training deferred values of the primitive items can be seen in Table 6.3. These schemas, developed in every run of the experiment, map out the actions needed to center a specimen in the camera, move towards it, and acquire it.

Learning Hidden Features

In every run, CLA used the lower-level schemas to create higher-level items that were not part of the original set of primitive items. For example:

```
[+s1/turnl/+s2].                                           (6.4)
```

Synthetic Item 6.4 is a synthetic feature that comes from Schema 6.1. The item indicates whether Schema 6.1 will succeed. In other words, when Synthetic Item 6.4 is true, then the robot can expect that turning left will move the specimen from Sector 1 to Sector 2. Because Schema 6.1 is unreliable, it is assumed that it will work only under certain conditions. Synthetic item 6.4 is designed to reflect those conditions. Synthetic items like these augment the original state space and add new properties. They became incorporated in higher-level schemas that further describe the robot's interaction with its environment, as seen in the next section.

Feature Refinement

The partitions s1 through s5 are low resolution, and would not be very useful for more subtle tasks. However, CLA recognizes that the blob tracker has more than five states and builds schemas that represent a specimen being in different places within a sector. For example:

```
+s1/turnl/+s1.                                             (6.5)
```

Schemas 6.5 states that sometimes turning left with a specimen in Sector 1 will result in the specimen remaining in Sector 1. This and related schemas were generated during every run of the ex-

periment. This can happen if the specimen shows up in the far left part of Sector 1. Schema 6.5 is an alternative to Schema 6.1: sometimes one applies, sometimes the other. The presence of both alternatives allows CLA to reason about both conditions, including how they relate to each other:

```
+[+s1/turnl/+s1]/turnl/+s1+[+s1/turnl/+s2]        (6.6)
```

Schema 6.6 says that, if the robot is in a state where turning left will keep the specimen in Sector 1, and then we turn left, then (of course) the specimen will be in Sector 1 *and* now the robot is in a state where turning left is more likely to move the specimen from Sector 1 to Sector 2. This schema was generated in 8 out of 10 runs.

By constructing synthetic items like these CLA is effectively refining the resolution of the blob tracker. Synthetic items not only represent new features, but also represent a refinement of existing features.

Object Persistence

CLA also learns that sometimes turning will make a specimen appear in the camera:

```
/turnr/+s5,                                       (6.7)
/turnl/+s1.                                       (6.8)
```

If the robot cannot see any specimens, Schemas 6.7 and 6.8 give it a policy to explore by turning. At the lower level, both of these schemas are roughly equal in reliability (Schema 6.7: M = .082, S.D. = .012; Schema 6.8: M = .091, S.D. = .014) $t(18) = -1.42$, $P > 0.5$ (two-tailed), and sometimes the robot will alternate between the two schemas, turning back and forth. When the second layer trains, however, Schemas 6.7 and 6.8 are used to create synthetic items which address the problem of the alternating strategies:

```
[/turnr/+s5]/turnr/[/turnr/+s5].                  (6.9)
```

Schema 6.9 says that if the robot is in a state such that turning right might reveal a specimen on the right side of the camera, and the robot turns right, then the robot can expect that turning right again reveals a specimen. Schema 6.9 (M = .314, S.D. = .052) is more reliable than Schemas 6.7, $t(18) = -14.032$, $P < 0.001$ (two-tailed), and 6.8, $t(18) = -13.365$, $P < 0.001$. The increased reliability of Schema 6.9 mean that it gets selected over Schemas 6.7 and 6.8, which encourages turning in the same direction rather than turning back and forth. Schema 6.9 was generated for every run.

CLA also uses these synthetic items at the second level to remember specimens that have moved out of view:

```
+s5/turnl/-s5+[/turnr/+s5].                       (6.10)
```

Schema 6.10 says that, if the robot sees a specimen at the far right, and the robot turns left, the specimen will disappear, but turning back to the right will make the specimen reappear. This schema was generated for every run. Taken together, Schemas 6.7 through 6.10 is the beginning of

the concept of object permanence. These schemas the robot the ability to find new specimens and reacquire lost ones.

CLA Performance

As stated earlier, the learning system acquired the necessary navigation schemas quite quickly. But the final piece of the puzzle was learning how to get the specimen in the grippers (Schema 6.3). Even though many schemas were learned in the first 1000 epochs, performance did not improve until Schema 6.3 was discovered, which usually occurred between 1500 and 3000 epochs. These basic schemas were enough to bring performance to a high level, which is not surprising considering how simple the domain is.

Upper-level schemas performed better than lower-level schemas. There are many possible explanations for the increase in performance, and this section describes one of these possibilities as an illustration of how higher-level schemas can improve performance.

The higher-level schemas that refine the blob tracker (described above) result in increased efficiency and can improve the robot's performance. At the lower level, CLA generated schemas like 6.1 through 6.3 to build a policy for obtaining the specimen. However, recall that the schema used most, Schema 6.2, is not very reliable. Sometimes moving forward would remove the specimen from Sector 3, if the specimen was to the left or right of center of Sector 3. These cases are described by their own schema:

```
+s3/forward/+s2.                                    (6.11)
```

Schema 6.11 states that sometimes when a specimen is centered, and the robot moves forward, the specimen will drift into Sector 2. This schema is less reliable than Schema 6.2 ($M = .113$, $S.D. = .020$), $t(18) = 23.46$, $P < 0.001$ (two-tailed), but nevertheless describes an alternative future state when approaching a specimen.

Should the specimen drift into Sector 2, the system can easily correct itself:

```
+s2/turnl/+s3.                                      (6.12)
```

Schema 6.12 states that a specimen in Sector 2 can be moved to Sector 3 by turning left. Before the upper-level schemas started training, these schemas gave a general strategy to acquire a specimen. However, in the case when the specimen is not centered in the camera's view, the strategy of these schemas has the robot arc towards the specimen — move forward until the specimen leaves the sector, turn left, repeat — rather than approach the specimen in a straight line (see Figure 6.4). When the robot is a long distance from the specimen, arcing increases the time to intercept the specimen. When the robot is closer, arcing increases the chance that the approach will be too oblique to get the specimen in the grippers, and will simply knock it away. There is no way, given the schemas at the first level, to solve this problem.

However, the upper-level schemas can identify the cases where lower-level schemas will succeed and fail. In fact, the very problem described above is captured by a second-level schema:

```
+s2/turnl/+s3+[+s3/forward/+s2].                    (6.13)
```

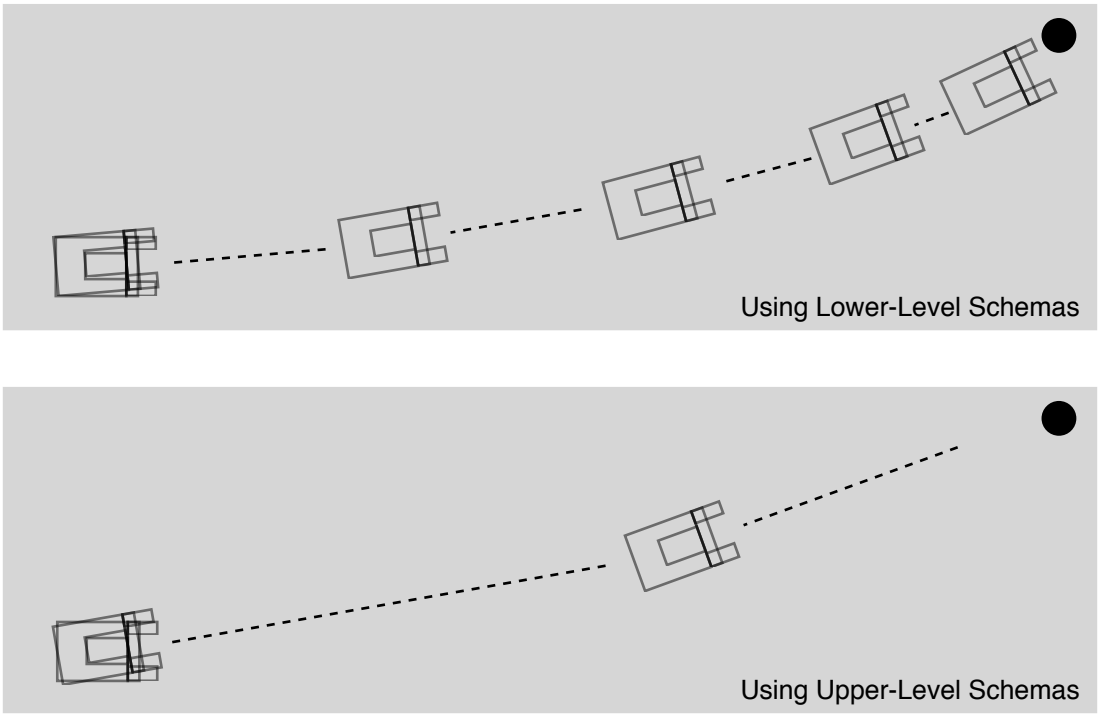

Figure 6.4: Two strategies for acquiring a specimen. Different strategies for acquiring a specimen are illustrated, using lower-level schemas (top) and upper-level schemas (bottom). The lower-level schemas had the robot rotate towards the specimen 5° at a time, increasing the time to the specimen and the chance that the approach would be oblique. Upper-level schemas built a strategy that accounted the increased time and turned the robot towards the specimen 10° at a time. The upper-level schema placed the specimen more in the center of the camera, decreasing the time to the specimen and the chance of an oblique approach which is less likely to succeed. Higher-level strategies such as these results in better performance than lower-level schemas.

Schema 6.13 says that if the specimen is in Sector 2, then turning left will center the specimen, but it also increases the likelihood that moving forward will put the schema back in Sector 2. This schema was generated for every run. This is not a schema that CLA can use to avoid this problem though because the result (s3) make Schema 6.13 appear valuable. However, another second-level schema describes how to fix the problem:

```
+s3+[+s3/forward/+s2]/turnl/+s3+[+s3/forward/+s3].
                                                    (6.14)
```

The context of Schema 6.14 is the same as the result of Schema 6.13. Schema 6.14 says, generally, that turning left twice will increase the chances that moving forward will keep the specimen centered. Schema 6.14 was generated in 7 of the 10 runs. Schemas 6.13 and 6.14 are very reliable (Schema 6.13: M = .88, S.D. = .002; Schema 6.14: M = .79, S.D. = .004), and will get chosen over 6.12 (M = .26; S.D. = .011). The result is that the robot, when centering on a specimen, will turn twice to get the specimen more in the center of the camera. The result is a shorter distance to the target, and less likelihood of an oblique approach (Figure 6.4). Higher-level schemas like these could account for the higher performance seen after 10000 epochs.

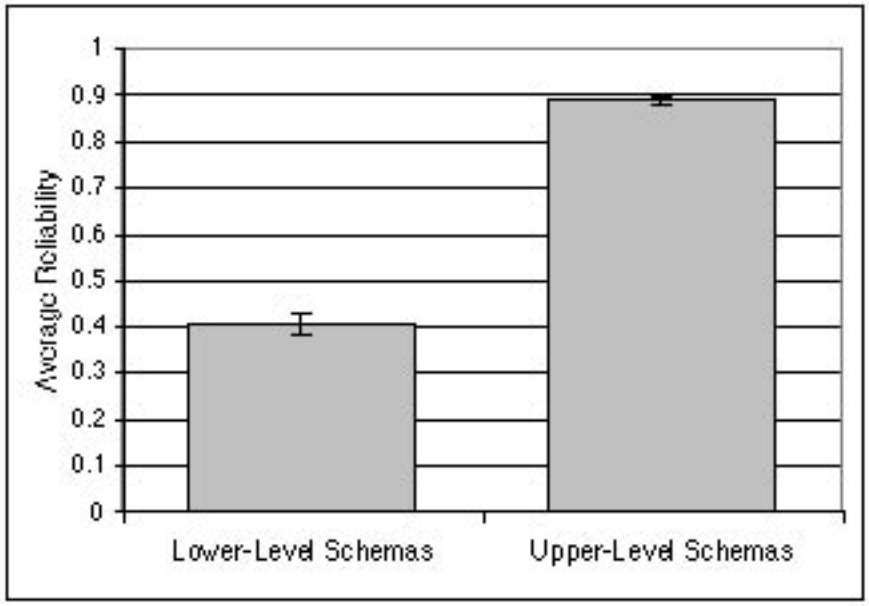

Figure 6.5: Average reliability of lower-level and upper-level schemas. Reliability of a schema is the percentage of times that the schema performed and the result was obtained. The error bars show the standard error. Lower-level schemas represent a coarse view of the environment, and are less reliable. Upper-level schemas are significantly more accurate and thus more reliable.

Schema Reliability

As noted earlier, higher-level schemas like 6.9 more reliable than lower-level schemas such as 6.7 and 6.8. This disparity is generally true of most lower-level and higher-level schemas. The final reliability (taken at the end of 20000 training epochs) of all first-level and second-level schemas was averaged over ten training runs. The results can be seen in Figure 6.5. First-level schemas had an average reliability of 40.5% (S.D. = .010), while second-level schemas had an average reliability of 88.6% (S.D. = .074). This difference is statistically significant, $t(18) = -13.389$, $P < 0.001$ (two tailed). A histogram of schema reliabilities for a typical single run is shown in Figure 6.6.

The increased reliability indicates that the schemas at the second level were a better model of the environment than the schemas at the first level. Second-level schemas were more reliable because they used synthetic items that more accurately reflected the environment. This particular experiment is a very simple task where the first-level schemas did quite well, so the difference was relatively small. But improved reliability increases the chance that a schema will get chosen, as shown earlier in this section.

6.2.7 Experiment 1 Conclusion

The foraging experiment showed that CLA can learn using delayed rewards. It improves on other Reinforcement Learning systems by building new features from the environment that can be used in a control policy. CLA developed a set of schemas that were used to allow a realistic robot to target and acquire specimens. Higher-level schemas were developed that gave the robot a more sophisticated and more reliable set of policies. The constructivist process resulted in new features that more accurately reflected the environment than the primitive features.

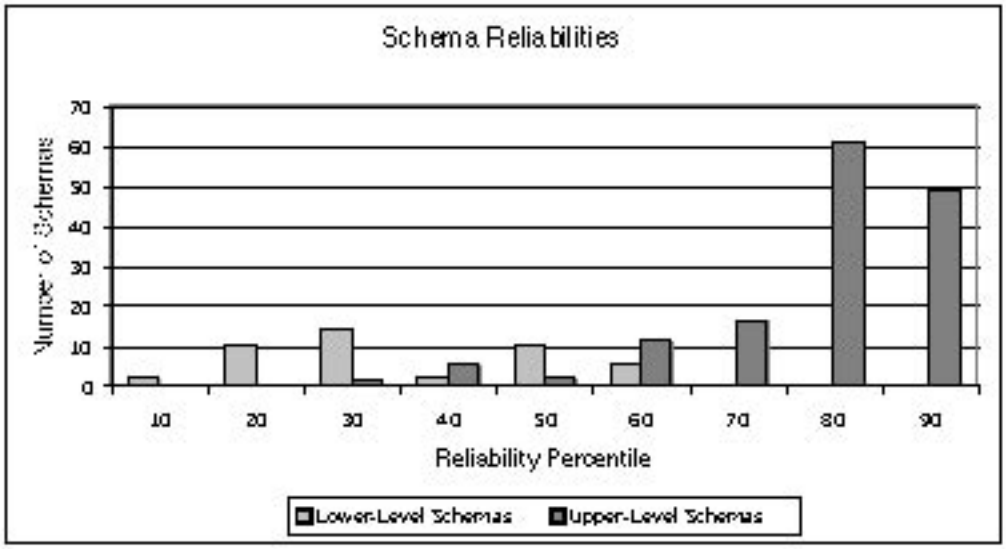

Figure 6.6: Histogram of schema reliabilities. This histogram shows the number of schemas in each reliability percentile for lower-level schemas (light gray) and upper-level schemas (dark gray) for a single typical run. Lower-level schemas averaged 41% reliability (S.D.. = 0.16) while upper-level schemas averaged 89% (S.D. = 0.11). Second-level schemas were more reliable because they used synthetic items that more accurately reflected the environment.

6.3 Experiment 2: Graceful Recovery from Damage

The previous experiment demonstrated that higher-level knowledge is useful. In this experiment, the role of the lower-level knowledge is underscored. In section 4.4, the importance of a hierarchy of knowledge was shown by demonstrating fallback when CLA was confronted with confusing input. Fallback allowed the knowledge base to process noisy stimuli by using lower-level knowledge — even when higher-level representations failed to process the stimuli. In experiment 2, fallback is shown for a mobile robot. The robot is shown to ability to degrade gracefully and operate robustly when its sensory apparatus is dramatically altered.

A trained robot is tested to determine its ability to recognize and process noisy data. In this case, the data is made noisy by "damaging" the robot, resulting in a 12° turn in the robot's camera. This change is meant to simulate the damage a robot might receive by running into an obstacle with some force, like a rock outcropping. Similar to the CLA model of causal perception in Chapter 4, the CLA foraging controller can process the altered information it receives at the lower level, even if the upper level fails to process the information.

6.3.1 Experiment Setup

The robot, using the trained knowledge bases from the previous experiement, had its camera rotated 12° to the left, which is equivalent to one visible sector (Figure 6.7). The robot was allowed to try and acquire specimens. During this time, no CLA training was performed, and reliability information was not altered at first. After performance was measured using the old reliability information, the system updated the reliability of schemas and the performance was measured again. This was done 10 times and the performance and reliability was obtained using the same technique as the last experiment.

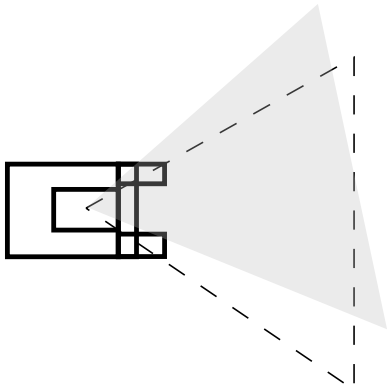

Figure 6.7: The damaged robot's visual system. The damaged robot's camera has been turned to the left 12°. It's current view is in gray. The original view is in the dotted lines for comparison. This change represents the damage a robot might sustain, change its view of the world. CLA provides the robot a way to recover from this damage.

6.3.2 Results

As expected, damaging the robot impacted the performance of the robot considerably. Without resetting any of the schema reliabilities or retraining any of the action maps, performance (which was measured the same way as in section 6.2.5) dropped to near zero (M = 1.6, S.D. = 0.84). As the reliability of the schemas adjusted, performance came back up to an average of 17.4 specimens collected (S.D. = 3.37; see Figure 6.8).

However, CLA was now using mostly lower-level schemas because of a big drop in reliability at the higher level. Lower-level schemas did lose reliability, dropping on average from 40.5% to 34.8% reliability (S.D. = .048), $t(18) = 3.689$, $P = 0.002$ (two-tailed). Higher-level schemas had a more pronounced drop in reliability, from 88.6% to 44.6% (S.D. = .032), $t(18) = 10.821$, $P < 0.001$ (two-tailed). These results are discussed in the next section.

6.3.3 Discussion

After the robot was damaged, the schemas that had been the most reliable become useless for guiding the robot towards the specimens. These schemas then adjusted their reliability to reflect the new state of the environment. The updated reliabilities allowed CLA to make more informed decisions regarding which schemas to use, and performance returned to near pre-damage levels.

But CLA had started using more lower-level schemas and neglecting the higher-level schemas because higher-level schemas had become much less reliable. The reliability of the schemas at both the first and second level were impacted. However, the impact for the second-level schemas was greater than that of the first-level schemas. The reliability of all lower-level and upper level schemas were averaged and compared to before-damage reliabilities (Figure 6.9). Lower-level schema reliability dropped from 40.5% to 34.8%, which is a difference of nearly 6 points, or 14% of peak performance. Several of the individual lower-level schemas dropped to near 10% reliability, mostly those with the forward action. But the average stayed high because some schemas became more reliable.

However the higher-level schemas became much less reliable. The reliability of upper-level schemas dropped from 88.6% to 44.6%, a drop of 44 points, 50% of peak performance. The reason for this difference is that the higher-level schemas are only as good as the lower-level schemas upon which they rely. Higher-level schemas are built using the most reliable schemas

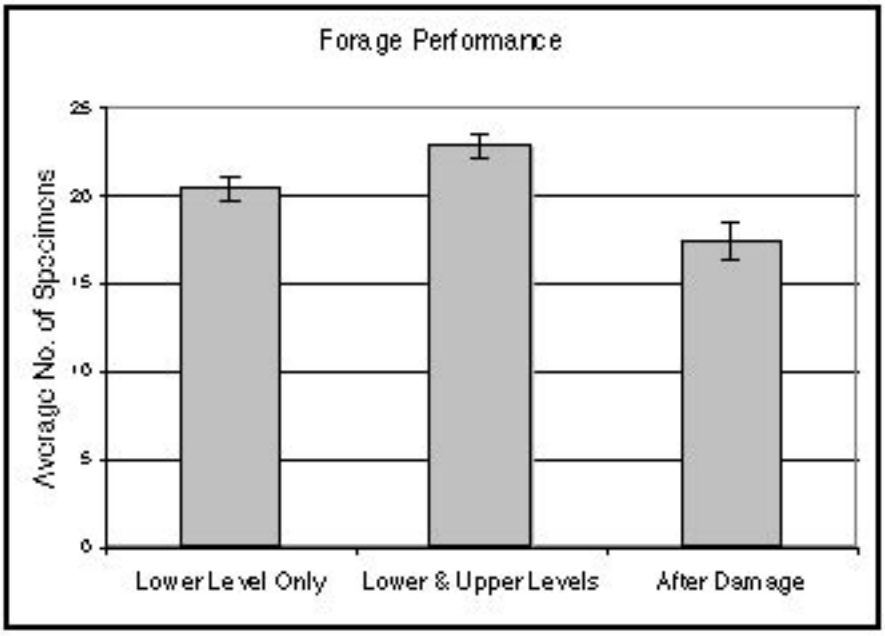

Figure 6.8: Performance of the robot at different stages. The robot's performance is measured using lower-level schemas before higher level schemas are trained, after the higher level schemas are trained, and after the robot received damage. The higher-level schemas slightly improved the performance of the robot. The damage initially brought performance to near zero. But as CLA adjusted the reliability of the schemas, the robot recovered most of its performance. This is an demonstration of fallback and recovery in a mobile robot.

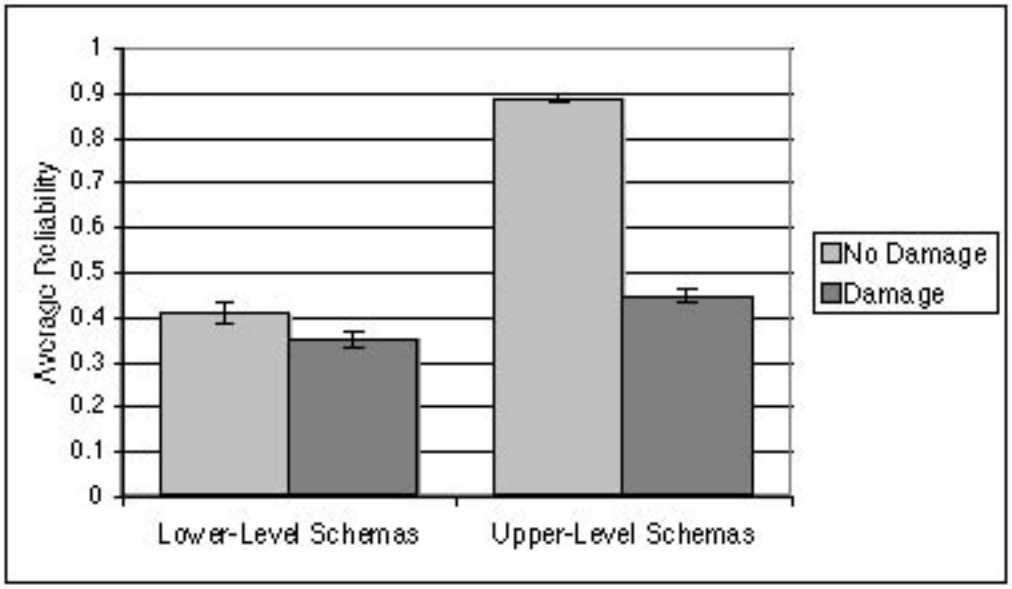

Figure 6.9: Average applicability of schemas before and after damage. After the damage to the robot's camera, the lower-level schemas drop from 40.55% reliability to 34.84%. But the higher-level schemas drop from 72.1% to 44.82%. Both changes are statistically significant. Higher-level schemas are more tuned to the environment and more brittle than the course lower-level schemas. Both the drop in performance and the recovery were statistically significant. The presence of the lower-level schemas allows the damaged robot to degrade gracefully by relying on lower-level schemas when the upper-level fails.

from the lower level. When these schemas become unreliable, then the higher-level schemas fail in greater numbers. For example, schema 6.2 becomes unreliable, then schema 6.14 — which offered such a great advantage before the damage — is now so unreliable that it is useless. Upper-level schemas have focussed on the most reliable lower-level schemas to build higher-level knowledge. When reliability at the lower-level shifts, it disappears at the higher-level.

This problem is not peculiar to CLA. The reason the higher-level schemas are failing is because they are a more accurate representation of the environment before the damage to the robot. Any highly accurate model will fail greatly when the assumptions of the model change. On the other hand, a coarse representation while less accurate is not so brittle. CLA solves the problem of trade-off between gross and fine-tuned models by building one from the other, and keeping them both. Maintaining the multiple levels of a hierarchical knowledge base allows CLA to be highly accurate under optimal conditions, and highly reliable under suboptimal conditions. This makes CLA preferable to any robot controller that is hand-coded and tailored to a given environment or task.

6.3.4 Experiment 2 Conclusion

Experiment 2 demonstrated that a robot using CLA as a controller will respond robustly to changes in the sensory information, including changes caused by damage to the robot. By updating the reliability of schemas alone, CLA can recover most of its pre-damage performance by falling back to lower-level schemas. While higher-level schemas increase by more accurately representing the environment, lower-level schemas also give CLA an advantage by providing a backup system under adverse conditions. CLA gives a robot grounded and robust autonomy.

6.4 Related Work

While CLA is unique in its approach to learning with delayed reward, there are several related technologies. These technologies — including Hierarchical Reinforcement Learning, Temporal Transition Hierarchies, Hierarchies of Abstract Machines, the Spatial Semantic Hierarchy, and learning from uninterpreted sensors — are compared to CLA below.

6.4.1 Hierarchical Reinforcement Learning

There is a type of reinforcement learning that builds a hierarchy of knowledge called Hierarchical Reinforcement Learning (HRL) (see Barto & Mahadevan 2003 for an overview). HRL starts as a standard Reinforcement Learning system by learning a global policy table for the environment. HRL then uses this table to explore and develop substructures of the global policy table. These substructures become higher-level "macro" actions that can by used to express a series of elemental actions. In the case of MAXQ (Deitterich 2000) and HEXQ (Hengst 2002), the state space is searched for common subtasks that are repeated or utilized in different parts of the policy table. Another example is Options (Stolle & Precup 2002; Kretchmar, Feil & Bansal 2003; Mannor et al 2004), which partitions the state space and identifies transitional subgoals to move from one partition to the other. In general, HRL is concerned with grouping actions together into a macro action.

CLA, on the other hand, is not building macro actions, but constructing higher-level features. The CLA approach to learning with delayed rewards is to refine the state space and build

new features to describe unseen states. These higher-level features are not simply the sum of their parts, but represent hypothetical features that were not prebuilt into the learning system. HRL's macro actions are not new in this way; instead, they are simply a collection of atomic actions.

6.4.2 Temporal Transition Hierarchies

Temporal Transition Hierarchies (Ring 1994) are very similar to the HRL systems described above. This system connects states to actions using a neural network architecture. The weights of the network represent the desirability of an action given a state, or the importance of a state following an action. As the system is trained, it locates state/action pairs with a high connecting weight and combines them into a "higher-level" unit that can be added to the network.

The central difference between Temporal Transition Hierarchies and CLA is the same as with HRL. The higher-level units learned in Temporal Transition Hierarchies are not novel or hypothetical states in the same way that CLA's schemas are. Instead, they are simply sequence of actions and state tests that are chained together and treated as a single unit. CLA brings more to Reinforcement Learning by introducing new features that augment and refine the state space.

6.4.3 Hierarchy of Abstract Machines

A learning system more akin with CLA is the Hierarchy of Abstract Machines (HAMs; Parr & Russell 1998). HAMs are a hierarchy of Reinforcement Learning systems that each operate over different state space. Each state space is a higher-level abstraction of the state space below it. The current state is represented within all multiple state spaces simultaneously, and the HAMs learn policies at these different levels of abstraction. This is very similar to the way CLA operates, except that with HAMs the ontological hierarchy is prebuilt. CLA builds its own ontology and learns actions using the features it has developed.

6.4.4 Spatial Semantic Hierarchy

The Spatial Semantic Hierarchy (SSH) (Kuipers 2000) is also a hierarchical ontology for robot control. The SSH ontology has four levels — control, causal, topological and metrical — that represent a robots environment. Observations at a lower level can be used to build knowledge at a higher level, while policies at a higher level will translate into motor commands at the lower level. CLA differs from the SSH in the same way that it differs from HAMs. The SSH is a prebuilt hierarchical ontology, while CLA builds its own hierarchical ontology.

6.4.5 Learning from Uninterpreted Sensors

CLA is probably most similar to work done in learning from uninterpreted sensors (Pierce & Kuipers 1997). This work involves a robot equipped with sensors, but the robot's control system does not know what these sensors represent. The sensor readings are correlated with motor commands to build an interpretation for each sensor and their relationship to each other. Once a sensor interpretation is built, the sensors can be used to perceive the environment and participate in action policies.

CLA also treats its primitive features as uninterpreted sensors. Schemas are CLA's way to building an interpretation for these features through correlated motor commands. Pierce and Kuipers (1997) uninterpreted sensors are not entirely uninterpreted, though, as they are assumed to represent the spatial features of the environment. CLA makes no such assumption.

However, the largest difference is that Pierce and Kuipers (1997) builds a single spatial interpretation of sensors and then terminates. CLA on the other hand continues to build new levels of features from the derived interpretations. In a way, CLA treats every level as a set of uninterpreted sensors, and uses it to construct yet another level of features.

6.5 Conclusion

This chapter demonstrated that CLA gives a robot robust autonomous control. CLA learned goal-directed behavior on a realistic robot and developed a hierarchy of sensorimotor schemas. The upper-level schemas refined the state space and gave the robot more sophisticated control. While the lower-level schemas were utilized when the robot was damaged, allowing the robot to recover most of its original performance.

CLA also represents a new approach to learning from delayed rewards. CLA is a complementary approach to Reinforcement Learning that elaborates the state space with higher-level features indicating the reliability of schemas. These higher-level features become incorporated in the learned policies and result in increased performance and higher reliability.

Finally, this chapter, along with Chapters 4 and 5, demonstrate the central hypothesis of this dissertation, that a model of infant cognitive development gives a mobile robot robust control.

7. Discussion and Future Work

In this Chapter, I discuss the accomplishments of the work in this dissertation, and suggest some ways in which this work can be advanced. First, I address implementation of CLA itself. The next section discusses achievements and potential advances in robotics. The third section views the work in the context of psychology. The fourth and final section suggests possible future work in neuroscience.

7.1 CLA Implementation

CLA, as presented in Chapter 3, is a simple and modular implementation of an unsupervised hierarchical learning system. It uses the SOM as a learning module for each layer. The layers are organized into levels, which learn a hierarchical representation of the environment on which CLA is trained. SOM layers can communicate with one another by using the trained nodes as new feature detectors and generating an activation vector. The activation vector re-represents the stimulus in terms of the higher-level schemas, and becomes input to another still-higher-level layer.

The implementation of CLA presented in this dissertation, in part, demonstrated that a simple hierarchical learning system can bring a number of benefits. But there are several aspects of CLA that have not been addressed, and improvements can be made in these areas. This section discusses some of these possible enhancements to the CLA implementation.

7.1.1 Recovery with Re-Learning

Part of the promise of the fallback abilities of CLA is not just that it degrades gracefully but also recovers robustly. In the event of an information overflow — either because of sensor damage, or changes in the environment, or some other unforeseen event — CLA will recognize that its higher-level representations are failing to assimilate the environment and drop to a lower level. In Chapter 6, partial recovery was achieved by determining which schemas were more reliable in the new world state, and which schemas should be ignored. The obvious next step is to re-learn the schemas to better fit the changed environment.

Presumably, this retraining should take less time than the original training. The environment is more likely to change incrementally than cataclysmically, and the existing schemas could easily shift from one representation to a another similar one. But CLA will still be able to correct

its representation even in the event of a radical change, even in the most extreme cases where it must re-randomize the schemas and start all over.

While the method of retraining by re-learning is straightforward, it may be difficult to decide when to re-learn. In all the applications presented in this dissertation, learning took place through random exploration. Indeed, in the early attempts to use CLA with a mobile robot in Chapter 6, replacing random exploration with 100% goal directed behavior reduced performance, as good techniques went undiscovered while the robot pursued mediocre techniques that happened to develop early in training. Additional logic will be required to determine whether the short-term drop in performance that comes from re-learning is worth performance increase in the long term.

7.1.2 Alternatives to the SOM

The SOM was chosen for CLA because of its ability to cluster stimuli and its neural plausibility. As was discussed in section 2.2, the SOM is not the only learning system that could work as a module for CLA. In theory, any clustering algorithm would suffice. The challenge will be to use the learning system to produce an activation vector in order to connect to a higher layer.

One possible alternative to the SOM would be Independent Component Analysis (ICA; Hyvärinen, Karhunen & Oja 2001). ICA could be used to build a set of independent features that describe the environment. Using ICA with CLA would likely result in a more compact representation than the SOM. Although there is much more computation needed for ICA than the SOM, smaller prototype vectors and reduced computational overhead would allow CLA to scale to more complex applications. CLA already improves over the Schema Mechanism for constructivist learning. Using ICA could make CLA still more efficient.

7.1.3 Alternative Integration Methods

While the SOM is good at building representations of cooccurring stimuli, it is not very good at recognizing other regularities in the environment, such as rotational, translational and scaling invariance. It is possible that some lower levels could eventually learn these regularities, but implementing more sophisticated feature detectors would make the learning process faster. In addition, some knowledge may require low-level integration methods; it is possible that such knowledge cannot be learned without rotational invariance, for example. A better understanding of this relationship between integration methods and learnability would help make CLA a more powerful learning system.

7.1.4 Using Neighboring Schemas

CLA, as used in the experiments in this dissertation, takes advantage of the SOM's ability to cluster stimuli into prototypes, and represent more frequent stimuli with finer detail. One feature of the SOM that is not used is the neighborhood effect, where neighboring nodes represent similar prototypes (section 2.2.2). This is information already present in CLA but currently unused.

One potential use of this information is to develop alternative interpretations (in the case of propositional schemas) or alternative strategies (in the case of sensorimotor schemas). In practical terms, neighboring schemas could provide an alternative strategy for sensorimotor schemas. If some goal was desired, and no schema had that goal as its result, CLA could search for schemas that resulted in items that are similar to the goal. Similarly, values that are assigned to schemas (as described in section 6.1.1) could be distributed to neighboring schemas, on the presumption that

they represent similar sensorimotor associations. The neighborhood feature of the SOM could be used for these and other improvements to the existing CLA algorithm.

7.1.5 Comparison to Reinforcement Learning

As discussed in section 6.1.1, CLA is a complementary approach to Reinforcement Learning that elaborates the state space, and addresses the problem of an underdefined feature set. A future experiment could demonstrate the importance of constructivism in reinforcement learning. This experiment would compare a standard Reinforcement Learning algorithm to one that used CLA on a task where the feature set is underdefined. The task and feature set could be similar to those used for the experiments in Chapter 6, with the addition of new obstacles to overcome, such as invisible barriers. Reinforcement Learning would have a problem with such a highly aliased environment, and may not be able to learn a sufficient policy acquiring specimens. CLA, however, would be able to build new features that described the barriers, and use these features to build strategies for avoiding them.

7.2 Robotics

Chapter 5 demonstrated that CLA is a more efficient implementation of the Schema Mechanism (Drescher 1991), building a hierarchy of sensorimotor schemas for use in an autonomous agent. Chapter 6 showed that CLA can move beyond the Microworld and control a realistic robot in a continuous environment. It is a new approach to learning with delayed rewards that brings constructivism to Reinforcement Learning. CLA is a robust robot controller that responds to damage and other changes by falling back to an earlier skill level rather than failing outright.

These experiments are just a beginning for applying CLA to robotics and there is great potential for future work to test the limits of CLA and expand its capabilities.

7.2.1 Physical Robot Applications

Controlling a simulated Pioneer robot is different from a controlling physical Pioneer robot. A physical robot controller has to deal with many issues: sensor noise and motor noise, moving on different surfaces, operating under different lighting conditions, and so on. As a robust learning system, CLA is uniquely appropriate for physical robot learning because it has the capacity to learn about these different conditions and integrate this knowledge into a goal-directed policy. If CLA can account for the issues related to physical robotics, it may be possible to avoid programming for these conditions altogether, and allow CLA to build schemas that can handle conditions like slippery floors or unreliable range sensors. Using CLA to handle these problems would help reduce the burden of the robot engineer.

7.2.2 Other Robot Platforms

CLA makes no assumptions about the sensory or motor apparatus that the robot has. This means that CLA is flexible enough to work with any robot configuration. The knowledge base generated by CLA on different robotic platforms would be useful in two ways. For one, the knowledge base would include schemas that reflect the peculiarities of the given robotic platform, and could be used as a starting point for robot engineers. If an online learning system is not tenable for

a given application, CLA could be used to pre-learn the knowledge base, and the robot engineers could use the resulting hierarchy of schemas to program the robot for more complex tasks. Like the idea of using CLA to handle peculiarities in the environment, CLA can be used to handle peculiarities in the robot apparatus.

Along the same lines, the knowledge could also be used as a diagnostic. By learning a knowledge base on a robot platform and analyzing the resulting knowledge hierarchy, engineers can see if CLA is compensating for flaws in the robot with corrective schemas. Since CLA's built knowledge reflects the agent, the knowledge can be used to analyze the agent.

7.3 Psychology

CLA is a model of infant cognitive development that can replicate infant studies. It is the first model of infant cognition that is also a mobile robot controller, and it brings all the features of constructivist learning — grounded knowledge, adaptation and robustness — to robotics. Using a cognitive model to control a robot expands our understanding of infant cognitive development, and gives an important modeling tool to the field of infant psychology.

While CLA has reproduced infant studies, it could also be used to make predictions of its own. It could also be used to apply the results of robotics applications back to infant cognition. Finally, CLA can be applied beyond infant cognition to higher-level cognition. These three topics are discussed in more detail below.

7.3.1 Testable Predictions

An important step in the acceptance of CLA as a model of infant cognition is to provide testable predictions. CLA must model some domain in infant cognition and provide a hypothesis that can be tested in the lab. If this is achieved, then CLA will be verified as a powerful model of infant cognition. Additionally, any feedback from these predictions can inform a future version of CLA, making CLA a better model and a more useful tool for robotics and cognition.

An obvious and excellent domain for CLA to generate testable hypotheses is motor development. This is discussed next.

7.3.2 Understanding Motor Development through Robotics

CLA is used to model causal perception in Chapter 4. Although it is controlling a robot in Chapter 6, CLA has not modeled any infant motor studies. Motor development in infants is an important part of cognitive development, and some have said it is vital for learning certain concepts such as distance perception and spatial search (Campos et al 2000). It is a logical extension of the work in this dissertation to bring what has been learned about motor control with CLA to robotics and relate it to CLA's model of infant cognitive development. For example, it may be possible to replicate the results of Campos et al (2000) by testing a robot's ability to visually search for items with a strictly propositional set of schemas (as in Chapter 4) versus a set of sensorimotor schemas (as in Chapter 6). This study could confirm that motor development is used for spatial search, as Campos et al (2000) has shown for infants.

A model of infant motor development would be an important contribution to infant cognition. It would also further mature CLA in both robotics and infant cognition. This kind of mutual

contribution is the power of a multidisciplinary computational model because, as each field answers one set of questions, it simultaneously poses a new set of questions to the other field.

7.3.3 Adult Cognition and Language

CLA may be applicable to cognitive modeling beyond infancy. The Information Processing Principles, upon which CLA is based, is postulated to account for learning throughout adulthood. CLA could be used to model adult cognition to demonstrate that human learning uses the same techniques at any age. Demonstrating this common learning system would be an important contribution to psychology, and would open the door for CLA to model a number of adult cognitive tasks.

Chief among these tasks is natural language processing. Language has long been the center of a debate on the origins of intelligence. CLA could be used to build a model of language that captures its hierarchical nature, from phonemes to words to phrases to sentences. Recent studies have found that there my be more information in the environment about language than originally thought. Saffran, Aslin and Newport (1996) have shown that infants can find word boundaries in artificial speech by learning the statistical relationship between neighboring speech sounds. Saffran, Senghas and Trueswell (2001) reported that older infants then package collections of words into phrases. This part-to-whole progression is exactly the kind of hierarchical structure that CLA can learn.

Saffran, Senghas and Trueswell (2001) also suggested that statistical regularities are not enough to learn meaning, and Marcus et al (1999) have challenged the statistical approach to language learning by showing that infants can learn abstract algebraic rules that are central to language syntax. It is not clear that the associative techniques of CLA are enough to learn algebraic rules, but rules might be handled by expanding CLA's information integration methods, as suggested in section 7.1.3.

Many learning systems have been used to learn some aspect of language or model language acquisition (including the SOM), but a hierarchical learning system like CLA has never been used to learn natural language. A study of language acquisition using CLA would be a major contribution to adult cognition, language and computer science.

7.5 Neuroscience

Although CLA is based on the SOM, which has been shown to model neural maps in the human brain, this dissertation does not depend on whether CLA itself is neurally plausible. However, by using the SOM as its central learning module makes CLA closer to an accurate model of neuroscience than learning systems that are unrelated to neurology. CLA was designed with neural plausibility in mind by utilizing few techniques that are neurally implausible. The hope was to design a learning system that could one day model neural structures in the brain, while remaining a cognitive model and a robot controller. A learning system that models not only cognition but neuroscience as well would be a powerful tool, and would help explain the connection between the two. Its connection to robotics makes it more powerful still, as robotics validates the model in the real world, showing that the model actually works. The work of turning CLA into a neural model lies in the hands of future scientists, and this section discusses some of the approaches that can be used to connect CLA to neuroscience.

7.5.1 Neurological Confirmation

While the SOM models neural maps, and maps are known to be connected to one another, there is no computational model for multiple levels of maps in the human brain. CLA uses Hebbian learning to interconnect the SOM layers, so both the layers and their connectors have a basis in neuroscience. CLA's inter-layer communication should be compared and contrasted to connections between neural maps in the brain. This comparison would be a major contribution to computer science and neuroscience, plus the differences between the two can be used to make CLA more realistic, increase its efficiency and broaden its applicability.

7.5.2 Testable Predictions

If CLA can be demonstrated to realistically model neurology, CLA should be used to produce testable predictions about how neurons activate and develop. For example, CLA could build a hierarchical representation of visual stimuli. This representation could be used to predict how neural maps in the brain would organize the same sensory information, and these predictions could be compared to the neural learning patterns in the brain for the same visual stimuli. Much like the testable predictions in infant cognition (section 7.4.1), a prediction from CLA that can be tested would isolate differences and similarities between the brain and CLA. Just as it is powerful to have a model that is used in both cognition and robotics, it would be more powerful still if the same model was applicable to neuroscience, allowing all three disciplines to inform each other.

7.5.3 Prefrontal Cortex for Decision Making

In an example of knowledge that can be brought from neuroscience to cognition and robotics, CLA's goal-directed behavior can be informed by a neural approach. Currently, CLA uses a system of value and reliability to choose between schemas for planning and action. CLA could expand its neural plausibility by using a neurological technique to perform this reasoning task. The prefrontal cortex of the human brain has been implicated in making value judgements and choosing among options (Damasio 1994). A neurological model of decision making in the prefrontal cortex, particularly the limbic system, could be combined with the schema development process of CLA to make a more fully neural model of cognitive development. A neural model of decision making would help bridge the gap between neuroscience and cognition, and would allow benefits from one field to be brought to the other, just as CLA currently benefits both infant cognition and robotics.

7.6 Conclusion

CLA is the first model of infant cognition used as a robot controller, and it has made contributions to both domains. These contributions are starting points for future work in robotics, cognition, and learning systems. CLA also has potential as a neural model. A model that addresses two or more disciplines is an important tool that allows knowledge to be shared and utilized across domains. CLA is such a model, and advances with CLA in one discipline will benefit them all.

8. Conclusions

This dissertation demonstrated that a computational model of infant cognitive development gives a robot robust autonomy. Constructivist learning in infants, as described by the Information Processing Principles (Cohen, Chaput & Cashon 2002), is modeled by the Constructivist Learning Architecture, or CLA. CLA is implemented by an interconnected hierarchy of SOMs. CLA can replicate studies of infant cognition, and can produce both propositional schemas and sensorimotor schemas. CLA gives a mobile robot robust autonomy. These contributions are reviewed by chapter below.

Chapter 2 introduced the foundations of CLA. The SOM (Kohonen 1997) was reviewed, which functions as a central learning component of CLA. The Information Processing Principles (Cohen, Chaput & Cashon 2002), a collection of principles that describe infant cognitive development for many domains and throughout infancy, were introduced. The Information Processing Principles form the design specification of CLA. Chapter 2 also introduced the idea of fallback, which allows a learning system to respond to confusing input by falling back to a lower level of knowledge, rather than failing completely. Fallback is the attribute of the Information Processing Principles that CLA uses to give robots robust autonomy.

Chapter 3 introduced CLA, a new self-organizing hierarchical learning system for modeling infant cognition and controlling a mobile robot. CLA is built using the SOM, and learns knowledge in multiple layers. A layer of knowledge is develops as a new set of feature detectors. These feature detectors can take a stimulus and produce an activation vector, which becomes the input to the next layer. CLA layers can be connected in a variety of architectures. CLA can also detect when input is confusing and invoke fallback by ignoring higher layers and utilizing lower layers.

Chapter 4 demonstrated that CLA is a model if infant cognitive development. CLA replicates a set of studies on infant causal perception. It not only learns causal perception, but learns it in the same way that an infant does: starting with the components of the event, and integrating them into a causal view. CLA exhibited stage-like development, just as infants to. CLA also demonstrated fallback when presented with a noisy launching event. CLA could process the noisy event using the lower-level schemas, even though the upper-level schemas failed to process the event. CLA is the first computation model of constructivist learning in infants, and the first full implementation of the Information Processing Principles.

Chapter 5 demonstrated that CLA could learn not only propositional schemas, but also sensorimotor schemas, which are crucial for controlling a robot. CLA also reproduced the functionality of the Schema Mechanism (Drescher 1991), and did so more efficiently.

Chapter 6 demonstrated that CLA gives a realistic robot robust autonomy. CLA was used to control a simulated Pioneer robot and learned to forage for specimens. CLA learned using delayed rewards, demonstrating that CLA is a constructivist approach to standard Reinforcement Learning techniques. CLA's hierarchical knowledge base refined the original state space, found hidden features, and supported fallback when the robot's vision system was damaged. CLA recovered from the damage by using lower-level representations, and the robot regained most of its pre-damage performance.

Together, these chapters show an approach to learning that is applicable to infant cognitive development and robust robot control. CLA's hierarchical learning capabilities offer improved performance or enhanced capabilities to several other machine learning systems and cognitive models. CLA bridges the gap between computer science and psychology, and provides a learning system that contributes to both and supports the transfer of science and technology from one discipline to the other.

Bibliography

Baillargeon, R. (1987). Object permanence in 3.5- and 4.5-month-old infants. *Developmental Psychology*, **23**, 655-664.

Baillargeon, R., Spelke, E. S., & Wasserman, S. (1985). Object permanence in five-month-old infants. *Cognition*, **20**, 191-208.

Barto, A. G. and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4), 341-379.

Bednar, J. A., and Miikkulainen, R. (2000a). Self-organization of innate face preferences: Could genetics be expressed through learning?. In (AAAI 2000), 117–122.

Bednar, J. A., and Miikkulainen, R. (2000b). Tilt aftereffects in a self-organizing model of the primary visual cortex. *Neural Computation*, 12(7), 1721–1740.

Bogartz, R. S., Shinskey, J. L., & Schilling, T. H. (2000). Object permanence in 5.5-month-old infants. *Infancy*, **1**, 403-428.

Bruce, J., Balch, T. and Veloso, M. (2000). Fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS'00).

Campos, J. J., Anderson, D. I.,Barbu-Roth, M. A., Hubbard, E. M., Hertenstein, M. J., and Witherington, D. (2000). Travel broadens the mind. *Infancy*, **1**, 149-219.

Cashon, C. H., & Cohen, L. B. (2000). Eight-month-old infants' perception of possible and impossible events. *Infancy*, **1**, 429-446.

Chaput, H. H. (2001). Post-Piagetian constructivism for grounded knowledge acquisition. In *Proceedings from the AAAI Spring Symposium on Grounded Knoweldge*.

Chaput, H. H. (2003). Constructivist Learning: A Neural Implementation of the Schema Mechanism. In *Proceedings of WSOM '03, The Workshop of Self-Organizing Maps*, Kitakyushu, Japan.

Chaput, H. H. and Cohen, L. B. (2001). "A model of infant causal perception and its development, " *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, 182-187.

Chomsky, N. (1968). *Language and Mind*. Brace & World, New York.

Cohen, L. B. (1998). An information processing approach to infant perception and cognition. In G. Butterworth and F. Simion (Eds.), *Development of Sensory, Motor, and Cognitive Capacities in Early Infancy: From Sensation to Cognition*. Sussex: Erlbaum (UK) Taylor & Francis.

Cohen, L. B. and Amsel, G. (1998). "Precursors to infants' perception of the causality of a simple event," *Infant Behavior and Development 21 (4)*, 713-732.

Cohen, L. B., Amsel, G., Redford, M. A. and Casasola, M. (1998). The development of infant causal perception. In A. Slator (Ed.), *Perceptual development: Visual, auditory and speech perception in infancy.* London: UCL Press (Univ. College London) and Taylor and Francis.

Cohen, L. B. and Cashon, C. H. (2000). Infant object segregation implies information integration, *Journal of Experimental Child Psychology*.

Cohen, L. B., & Cashon, C. H. (2001). Do 7-month-old infants process independent features or facial configurations? *Infant and Child Development*, **10**, 83-92.

Cohen, L. B. & Cashon, C. H. (2003). Infant perception and cognition. In R. Lerner, A. Easterbrooks, and J. Mistry (Eds.), *Comprehensive handbook of psychology. Volume 6, Developmental Psychology. II. Infancy.* 65-89. New York: Wiley and Sons.

Cohen, L. B. and Chaput, H. H. (2002). Connectionist models of infant perceptual and cognitive development. *Developmental Science*, 5:2, 173-175.

Cohen, L. B., Chaput, H. H. and Cashon, C. H. (2002). "A constructivist model of infant cognition," *Cognitive Development*, **17**, 1323-1343.

Cohen, L. B. and Oakes, L. M. (1993). How infants perceive a simple causal event. *Developmental Psychology, 29*, 421–433.

Cohen, L. B. and Younger, B. A. (1984). Infant perception of angular relations. *Infant Behavior and Development, 7*, 37-47.

Damasio, A. R. (1994). *Descartes' Error: Emotion, Reason and the Human Brain*. New York: Putnam.

Dietterich, T. G. (2000). An Overview of MAXQ Hierarchical Reinforcement Learning. In B. Y. Choueiry and T. Walsh (Eds.) *Proceedings of the Symposium on Abstraction, Reformulation and Approximation SARA 2000, Lecture Notes in Artificial Intelligence*, New York: Springer Verlag.

Drescher, G. (1991). *Made-up minds: a constructivist approach to artificial intelligence*, Cambridge, MA: MIT Press.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, **14**, 179-211.

Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D. and Plunkett, K. (1996). *Rethinking innateness: a connectionist perspective on development*. MIT Press.

Fahlman, S. E., and Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2* (pp.524–532). Los Altos, CA: Morgan Kaufmann.

Farkaš, I and Li, P. (2001). A self-organizing neural network model of the acquisition of work meaning. In *Proceedings of the 4th Internation Conference on Cognitive Modeling, Fairfax*, VA, 67-72.

Gerkey, B., Støy, K., and Vaughan, R. T. (2000). "Player Robot Server". Technical Report IRIS-00-392, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Gerkey, B., Vaughan, R. T. and Howard, A. (2003). "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems" *Proceedings of the 11th International Conference on Advanced Robotics*, pp317-323, Coimbra, Portugal, June 2003.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. In C. Sammut and A. Hoffmann (Eds.), *Proceedings of the Nineteenth International Conference on Machine Learning* ICML 2002, 243-250.

Horowitz, A. L. (1995). *MRI Physics for Radiologists: A Visual Approach (3rd ed.)*. New York: Springer-Verlag.

Hume, D. (1777/1993). *An enquiry concerning human understanding*. Indianapolis, IN:Hackett.

Hyvärinen, A., Karhunen, J. and Oja, E. (2001). *Independent Component Analysis*. New York: J. Wiley.

Johnson, S. P., & Nañez, J. E. (1995). Young infants' perception of object unity in two-dimensional displays. *Infant Behavior and Development*, **18**, 133-143.

Kant, I. (1794/1982). *Critique of pure reason*. Translated by Wolfgang Schwarz.Aalen:Scientia.

Kellman, P. J., & Spelke, E. S. (1983). Perception of partly occluded objects in infancy. *Cognitive Psychology*, **15**, 483-524.

Kohonen, T. (1988). *Self-organization and associative memory* (2nd ed.). New York: Springer-Verlag.

Kohonen, T. (1997) *Self-organizing maps*, Berlin: Springer-Verlag.

Kretchmar, R. M., Feil, T., and Bansal, R. (2003). Improved automatic discovery of subgoals for options in hierarchical reinforcement learning. *Journal of Computer Science and Technology*, 3, no. 2.

Kuipers, B. J. (2000). The spatial semantic hierarchy. *Artificial Intelligence* **119**: 191-233.

Labiouse, C. L. and French, R. M. (2001). A connectionist encoder model of social perception and stereotyping. In R. French & J. Sougné (Eds.), *Proceedings of the sixth Neural Computation and Psychology Workshop: Evolution, Learning, and Development*, 208-219. London: Springer Verlag.

Leslie, A. M. (1984). Spatiotemporal continuity and the perception of causality in infants. *Perception, 13*, 287–305.

Leslie, A. M. and Keeble, S. (1987). Do six-month-olds perceive causality? *Cognition, 25*, 265–288.

Lewis, J. D. and Elman, J. L. (2001). A connectionist investigation of linguistic arguments from the poverty of the stimulus: learning the unlearnable. In J.D. Moore and K. Stenning (Eds.) *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society*, Mahwah, NJ: Lawrence Erlbaum Associates. 552-557.

Mannor, S., Menache, I., Hoze, A. and Klein, U. (2004). Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the 21st Internation Conference on Machine Learning*, Banff, Canada.

Marcus, G. F. (2002). The modules behind the learning. *Developmental Science*, 5:2, 175-176.

Marcus, G. F., Vijayan, S., Bandi Rao, S., Vishton, P. M. (1999). Rule Learning by Seven-Month-Old Infants. *Science* **283**: 77-80.

Mareschal, D. and Johnson, S. P. (2002). Learning to perceive object unity: a connectionist account. *Developmental Science* 5:2, 151-185.

Mareschal, D., Quinn, P. C. and French, R. M. (2002) Asymmetric interference in 3- to 4-month olds' sequential category learning. *Cognitive Science*, **26**, 377-389.

Michotte, A. (1963). *The perception of causality.* New York: Basic Books.

Munakata, Y. (2003). Computation cognitive neuroscience of early memory development. *Developmental Review*, **24**, 133–153.

Munakata, Y. and Pfaffly, J. (2004). Hebbian learning and development. *Developmental Science*, 7:2, 141-148.

Munakata, Y. and Stedron, J. M. (2002). Modeling infants' perception of object unity: what have we learned? *Developmental Science*, 5:2, 176-178.

Parisi, D. and Schlesinger, M. (2002) Artificial life and Piaget. *Cognitive Development*, **17**, 1301-1321.

Parr, R. and Russell, S. (1998). Reinforcement learning with hierarchies of machines. In *Proceedings on the Neural Information Processing Systems (NIPS) Conference*.

Piaget, J. (1936). *The origins of intelligence in children*, New York: International Universities Press.

Piaget, J. (1937). *The construction of reality in the child*. New York: Basic Books.

Pierce, D. and Kuipers, B. J. (1997). Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* **92**: 169-229.

Raijnmakers, M. E. J. and Molenaar, P. C. M. (2004). Modeling developmental transitions in adaptive resonance theory. *Developmental Science* 7:2, 149-157.

Ring, M. B. (1994). *Continual Learning in Reinforcement Environments*. PhD thesis. University of Texas at Austin.

Rumelhart, D. E. and McClelland, J. L. (1986). On Learning the Past Tenses of English Verbs. In Volume 2 of Rumelhart, McClelland, and the PDP Research Group (1986), pp. 216-271.

Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volumes 1 and 2*. Cambridge, MA: MIT Press.

Saffran, J.R., Aslin, R.N., & Newport, E.L. (1996). Statistical learning by 8-month-old infants. *Science*, **274**, 1926–1928.

Saffran, J. R., Senghas, A., Trueswell, J. C. (2001). The acquisition of language by children. *Proceedings of the National Academy of Science, U. S. A.* 98: 12874-12875.

Schilling, T. H. (2000). Infants' looking at possible and impossible screen rotations: The role of familiarization. *Infancy*, **1**, 389-402.

Schlesinger, M. (2004). Evolving agents as a metaphor for the developing child. *Developmental Science* 7:2, 158-164.

Schlesinger, M. and Barto, A. (1999). Optimal control methods for simulating the perception of causality in young infants. In M. Hahn & S. C. Stoness (Eds.), Proceedings of the 21st Annual Conference of the Cognitive Science Society, 625-630.

Schlesinger, M. and Parisi, D. (2001). The agent-based approach: A new direction for computational models of development. *Developmental Review*, **21**, 121-146.

Shultz, T. R. (2003). *Computational developmental psychology*. Cambridge, MA: MIT Press.

Shultz, T. R., & Bale, A. C. (2001). Neural network simulation of infant familiarization to artificial sentences: Rule-like behavior without explicit rules and variables. *Infancy*, **2**, 501-536.

Shultz, T. R. and Mareschal, D. (1997). Rethinking innateness, learning and constructivism: connectionist perspectives on development. *Cognitive Development*, **12**, 563-586.

Shultz, T. R., & Rivest, F. (2001). Knowledge-based cascade-correlation: Using knowledge to speed learning. *Connection Science*, **13**, 43-72.

Sirois, A. (2004). Autoassociator networks: insights into infant cognition. *Developmental Science*, 7:2, 133-140.

Sirois, S., Buckingham, D. and Shultz, T. R. (2000). Artificial grammar learning by infants: an auto-associator perspective. *Developmental Science*, 3:4, 442-456.

Slater, A. M., Morison, V., Somers, M., Mattock, A., Brown, E., & Taylor, D. (1990). Newborn and older infants' perception of partly occluded objects. *Infant Behavior and Development*, **13**, 33-49.

Smith, L. (2002). Teleology in connectionism. *Developmental Science*, 5:2, 178-180.

Spelke, E. S., Breinlinger, K., Macomber, J., & Jacobson, K. (1992). Origins of knowledge. *Psychological Review*, **99**, 605-632.

Steels, L. (2003) Evolving grounded communication for robots. *Trends in Cognitive Science*. 7(7), July 2003, 308-312.

Stolle, M. and Precup, D. (2002). Learning options in reinforcement learning. In S. Koenig and R. Holte (Eds.): *Proceedings of the 5th International Symposium of Abstraction, Reformulation and Approximation (SARA) 2002*, 196-211.

Stone, P. (2000) *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*, MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, Massachusetts: MIT Press.

Vaughan, R. T. (2000). Stage: A Multiple Robot Simulator. Technical Report IRIS-00-394, Institute for Robotics and Intelligent Systems, School of Engineering, University of Southern California.

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, 8, 279–292.

Westermann, G. and Mareschal, D. (2004). From Parts to Wholes: Mechanisms of Development in Infant Visual Object Processing. *Infancy*, **5**, 131-151.

Wilcox, T. & Baillargeon, R. (1998). Object individuation in infancy: The use of featural information in reasoning about occlusion events. *Cognitive Psychology*, **37**, 97-155.

Witkowski, M. (1998) Dynamic Expectancy: An Approach to Behaviour Shaping Using a New Method of Reinforcement Learning, SIRS-98, 6th International Symposium on Intelligent Robotic Systems '98, Edinburgh, Scotland, UK, July 21-23, 73-81.

Yoshikawa, Y., Koga, J., Asada, M. and Hosoda, K. (2003). A Constructive Model of Mother-Infant Interaction towards Infant's Vowel Articulation. In Prince, C. G., Berthouze, L., Kozima, H., Bullock, D., Stojanov, G. and Balkenius, C. (Eds.), *Proceedings Third International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems 101*, 139-146.