Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning

by
Leonard Newton Foner

SB EECS
Massachusetts Institute of Technology
May 1986

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in Partial Fulfillment of the requirements of the degree of

MASTER OF SCIENCE
in
MEDIA ARTS AND SCIENCES
at the
Massachusetts Institute of Technology
June 1994

Signature of Author _____

Program in Media Arts and Sciences
May 6, 1994

Certified By _____

Pattie Maes
Assistant Professor, Media Arts and Sciences
Program in Media Arts and Sciences

Accepted by _____

Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning

by
Leonard Newton Foner

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 6, 1994
in Partial Fulfillment of the requirements of the degree of

MASTER OF SCIENCE
in
MEDIA ARTS AND SCIENCES
at the
Massachusetts Institute of Technology

Abstract

Adaptive autonomous agents have to learn about the effects of their actions so as to be able to improve their performance and adapt to long term changes. The problem of correlating actions with changes in sensor data is $O(n^2)$ and therefore computationally expensive for any non-trivial application. I propose to make this problem more manageable by using focus of attention. In particular, I discuss two complementary methods for focus of attention: *perceptual selectivity* restricts the set of sensor data the agent attends to at a particular point in time, while *cognitive selectivity* restricts the set of internal structures that is updated at a particular point in time. I present results of several implemented algorithms—variants of the *schema mechanism* [Drescher 91]—which employ these two forms of focus of attention. The results demonstrate that incorporating focus of attention dramatically decreases the computational expense of learning action models without affecting the quality of the knowledge learned, with only small increases in the number of training examples required to learn the same knowledge.

Thesis Supervisor: Pattie Maes
Title: Assistant Professor, Media Arts and Sciences

This work was supported in part by NSF grant number IRI-92056688.

Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning

by

Leonard Newton Foner

The following people served as readers for this thesis:

Reader

Gary Drescher
Research Scientist
Thinking Machines Corporation

Reader

Marvin Minsky
Toshiba Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Acknowledgments

I would like to thank my advisor, Pattie Maes, for taking her duty as an advisor seriously. She has provided invaluable feedback, criticism, and advice, and has been an excellent role model.

Gary Drescher, in addition to serving as a reader of this thesis, implemented the basic machine learning system upon which this research was based. His basic system architecture, and his comments and feedback at several phases of the research, helped to get this research off the ground.

Marvin Minsky has, as usual, given me a lot to think about, in what he says and what he writes.

Robert Ramstad wrote a serial implementation of Drescher's original parallel system. This serial implementation was the jumping-off point for an essentially complete rewrite to serve the needs of this research. While just about every nail, board, and piece of rope in his original sailboat was eventually replaced (while simultaneously getting a superstructure, an outboard motor, a radar, three lifeboats, several sails, and a big fishing net attached), it's still a nice sailboat.

Bruce Blumberg's Hamsterdam system was the second domain microworld used in validating certain results, and he was very helpful in helping me to navigate through his code, and in pointing me at some useful resources for tunnelling through the wall between his C code and my Lisp code.

Cecile Pham wrote a graphical representation of the eyehand microworld in Macintosh Common Lisp that was used to demonstrate certain aspects of an early version of this research.

Jeremy Brown provided invaluable clues in unraveling some of the obscurities of the ALIVE puppet, despite the fact that I was asking him for help with something that he hadn't seen in months and which was half somebody else's code.

Elizabeth Manicatide reviewed various drafts of pieces of this thesis, was indispensable in providing me with tools, training, and expertise in graphics arts, and drew funny cartoons on every available surface.

Laura Dent provided many useful comments early on in the proposal process.

Finally, I would like to thank my family, who have always given me all the support I needed and have always been there for me.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 What is an adaptive autonomous agent?

An *agent* is a system that tries to fulfill a set of *goals* while *situated* in a complex and changing *environment*. The agent might be composed of hardware (e.g., a *robot*) and situated in the physical world, or composed entirely of software running in a computer. In the latter case, the agent may be situated in a simulated world (a *synthetic actor*) or may interact as a peer with other entities, such as network databases (a *software agent*, *interface agent*, *knowbot*, etc).

Being situated in this environment, the agent can sense it in various ways, and can take actions to change the environment or its place in it. The goals can be of many forms, such as *end goals* or goals of *attainment* (e.g., in a robot, finding a coffee cup); goals of *homeostasis* (e.g., not letting the robot's batteries run down); they may be *rewards* of some sort that the agent attempts to maximize or punishments that it attempts to minimize; and so forth.

The agent is *autonomous* if it operates in an independent fashion: in other words, when it decides itself how to relate sensor data to actions in a way that leads to timely or reliable satisfaction of its goals. The agent is *adaptive* if it can improve over time, presumably by learning.[1]

1. Other kinds of adaptation are certainly possible (for example, in biological systems, muscles adapt to repeated high loads by gradually increasing in strength). However, we will confine our attention here to *cognitive* adaptation—those techniques which allow the agent to *understand what to do better.*

Unless otherwise specified, the term *agent* in this thesis will be taken to be an adaptive, autonomous agent, whether physically-based or composed completely of software.

Given the specifications above, an adaptive autonomous agent thus has at least two major problems facing it:

- *Action selection*, in which it must decide what action to take next, and
- *Learning from experience*, in which it must improve its performance over time.

Neither of these problems is well understood; a summary of current open problems and progress to date appears in [Maes 94]. Current solutions to either problem tend to scale poorly, may have performance characteristics that are difficult to predict, can be difficult to reuse in different systems, can get stuck in behavioral loops, and more.

For agents that learn, [Maes 94] specifies some desiderata that should be addressed:

- Learning should be incremental, with the agent learning after every experience, rather than being divided up into separate learning and performance phases.
- The agent should be biased toward learning information which is relevant to its goals.
- Learning should be able to cope with a nondeterministic world, in which unpredictable things might happen occasionally, sensor information is noisy, and so forth.
- The learning should be unsupervised: the agent should learn mostly autonomously.
- Ideally, it should be possible to build in some knowledge to the agent at the start, so it does not have to start from scratch, especially in situations in which prior knowledge is easily available.

Additionally, she points out the problems that must be addressed when designing the architecture of a learning agent:

- How does the *action selection* mechanism work?
- How does the agent *learn*? What hypotheses can it create, and how does it decide which are worthwhile?
- What is the agent's *experimental strategy*? In other words, through what mechanism does the agent decide when to *exploit* (performing some task as optimally as it current knows how to do) versus when to *explore* (performing some action suboptimally in an attempt to discover a new, even better strategy).

This thesis primarily explores the question of learning. Along the way, it also investigates certain topics related to action selection and experimental strategy.

1.2 Some basic concepts

In studying learning in an autonomous agent, there are a few basic concepts that must be understood. First of all, any agent operates in some particular *world*. The characteristics of this world exert a strong influence on the design of the agent. For example, if the agent is operating in a very dangerous, physical world (such as exploring a rock face near a cliff), architectures which put great emphasis on accurate sensing and avoiding risk are quite important. On the other hand, a software agent investigating the contents of databases may emphasize exploratory behavior over most other considerations. The world also strongly determines what sorts of sensors the agent may have, what sort of data it can expect to receive from them, and so on. The design of the sensors and their interaction with the world may determine whether the world appears essentially deterministic or highly nondeterministic; this may in turn influence the design of the learning system, since not all learning sys-

tems can tolerate noise in their inputs, and some tolerate different kinds of noise in different ways.

The agent's *goals* also play an important role in its design. For example, can the agent choose which goal to pursue next, or is it directed externally? Are its goals primarily goals of attainment or goals of homeostasis? How many tactical (short-range) goals might it have to execute to reach a strategic (long-range) goal? (The latter question may determine whether the robot must engage in sophisticated reasoning or planning, for instance.)

Finally, when talking about learning in agents, one must decide whether the agent is to display any *selectivity* or *focus* in what it learns, or whether it should attempt to learn indiscriminately. In evaluating how well the agent is learning, one must ask about the *correctness* of the information learned—is the agent learning things that are actually true in the world?—and its *completeness*—is the agent learning enough? In the case of an agent which uses some form of selectivity or focus, one might also ask about *relative completeness*, in which the influence of its selectivity is considered: if the agent is only supposed to be learning about certain topics, one should restrict one's evaluation of its performance to those topics, rather than inquiring about its ability to learn *everything* about the world. In agents that have goals, one might also ask about the *relevance* of its learning to the performance of its goals: in other words, is what the agent learns useful in accomplishing its goals, and does it avoid trying to learning things that are not useful for those goals?

1.3 Focus of attention in learning

Autonomous agents have to learn about their environment so as to improve (because user programming has its limitations) and adapt (because things change). Several learning methods for autonomous agents have been proposed, in particular reinforcement learning [Sutton 91] [Kaelbling 93], classifier systems [Holland 86] [Wilson 85], action model

learners [Drescher 91] [Maes 92] and mixed methods [Sutton 90] [Booker 88]. No matter which of these algorithms is used, a learning agent will have to correlate some number of sensory inputs with some number of internal structures (its internal memory of what it has learned so far) in an attempt to extend its knowledge. This is conceptually a cross-product problem: each sensory bit should be correlated in some fashion with each already-built internal structures. As the number of sensory bits or the number of internal structures grows, the work required to perform this correlation grows approximately as $O(n^2)$. Solutions which can decrease either the constant or, preferably, the exponent, may be very helpful in keeping the work of learning within feasible bounds.

Most unsupervised learning algorithms attempt to learn all that there is to know about the environment, with no selectivity save the implicit or explicit limits on the generalizations that they can entertain.[2] They flail about, often at random, attempting to learn every possible correlation. It takes them far too long to learn a mass of mostly-irrelevant data. For example, the *schema mechanism* [Drescher 91] introduces an algorithm for building successively more reliable and abstract descriptions of the results of taking particular actions in an unpredictable world. However, the algorithm scales poorly, and hence is unsuitable for realistic worlds with many facts, given the current state of computational hardware. If no provision is made to bound the number of concepts that may be learned,[3] its running speed decreases monotonically as more is learned about the world. This means that, on currently-available hardware, the algorithm eventually becomes extremely slow.

2. For any finite set of data, there are infinitely many distinct hypotheses that are consistent with those data. However, all learning systems impose selectivity on the generalizations that they can entertain, whether that selectivity is implicit or explicit, by virtue of their representations of the domain and the operations they perform upon those representations.

3. Say, by assuming a finite learning lifetime, or by implementing some sort of garbage-collection of concepts that maintains a fixed upper bound on their number.

In general, real creatures use various *focus of attention* mechanisms, among others *perceptual selectivity* and *cognitive selectivity*, to guide their learning. By focusing their attention to the important aspects of their current experience and memory, real creatures dramatically decrease the perceptual and cognitive load of learning about their environment and making decisions about what to do next [Aloimonos 93]. This research uses similar methods of selectivity to build a computationally less expensive, unsupervised learning system that might be suitable for use in an autonomous agent that must learn and function in some complex world.

This thesis presents a range of algorithms for learning statistical action models which incorporate perceptual and cognitive selectivity. In particular, it discusses several variations on Drescher's schema mechanism [Drescher 91] and demonstrates that the computational complexity of the algorithm can be significantly improved without harming the correctness and relative completeness of the action models learned. The particular forms of perceptual and cognitive selectivity that are employed represent both domain-dependent and domain-independent heuristics for focusing attention that potentially can be incorporated into many learning algorithms.

The thesis is organized as follows. Chapter 2 discusses different notions of focus of attention, concentrating on heuristics for perceptual and cognitive selectivity, and lays a basic framework for this research. It also describes the two microworlds selected in which various algorithms were tested. Chapter 3 describes the basic methods of goal-independent focus of attention. It also introduces some notation for talking about the methods and their results, and discusses results in using goal-independent focus of attention. Chapter 4 then discusses the implementation of goal-directed focus of attention, extending the results presented in Chapter 3, and describes in detail some additional methods of evaluating the learning. Chapter 5 discusses related work in machine learning and cognitive science; both

fields have different insights into focus of attention. Chapter 6 summarizes the research, discusses certain limitations of the approaches used, and speculates on some possible future directions. Appendix A provides some details of the architecture used to run the learning systems, and Appendix B details the mechanisms used, in evaluating the learning in Chapter 4, to keep the planning process computationally efficient.

Chapter 2: The Basic Framework

2.1 Focus of Attention Methods

2.1.1 Introduction

To ease its learning task, an agent can employ a range of methods for focus of attention. It can be selective in terms of what sensor data it attends to as well as what internal structures it considers when acting and learning. These forms of focus of attention are termed *perceptual* and *cognitive selectivity* respectively. They are illustrated by the left and right braces respectively in Figure 1 below, and discussed in more detail in the following sections.

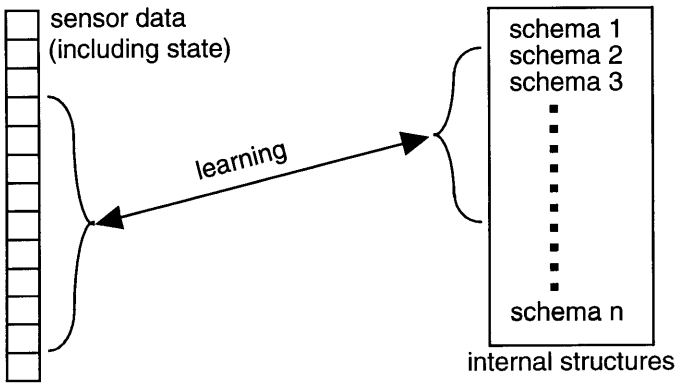

Figure 1: Sensory (left) and cognitive (right) pruning

Along another dimension, there is a distinction between *domain-dependent* and *domain-independent methods* for focus of attention. Domain-independent methods represent general heuristics for focus of attention that can be employed in any domain. For example, one can attempt only to correlate events that happened close to one another in

time. Domain-dependent heuristics, on the other hand, are specific to the domain at hand. They typically have been preprogrammed (by natural or artificial selection or by a programmer). For example, experiments have shown that when a rat becomes sick to its stomach, it will assume that whatever it ate recently is causally related to the sickness. That is, it is very hard for a rat to learn that a light flash or the sound of a bell is correlated to the stomach problem because it will focus on recently eaten food as the cause of the problem [Garcia 72]. This demonstrates that animals have evolved to pay attention to particular events when learning about certain effects.

Finally, the focus mechanism can be *goal-driven* and/or *world-driven*. Focus of attention in animals is both strongly world- and goal-driven. The structure of the world and the sensory system determines which sensory or memory bits may be "usually" ignored (e.g. those not local in time and space), while the task determines those which are relevant some of the time and not at other times. For example, when hungry, any form of food is a very important stimulus to attend to; learning how to get to some would presumably take on greater important in this case.

The results reported in Chapter 3 concern world-dependent, domain-independent cognitive and sensory selectivity. Such pruning depends on invariant properties of the environment[1] and common tasks, and does not take into account what the current goal of the agent is. The methods can be applied to virtually any domain. While it is true that, in complex worlds, goal-driven and domain-dependent pruning is quite important, it is surprising how much of an advantage even goal-independent pruning can convey. Using Chapter 3's work as a base, Chapter 4 then investigates the added leverage of adding goals to the learning

1. In a physical, terrestrial environment, such properties might include causality (actions must precede their effects), locality (most effects are near their causes), space, time, gravity, etc.

system. That is, the actions taken and cognitive and perceptual pruning that occurs are controlled by the short-term goals of the agent.

2.1.2 Perceptual selectivity

Perceptual selectivity limits what stimuli might possibly be attended to at any one time, which puts limits on what might be learnable at that time. For example, a real creature would not pay attention to every square centimeter of its skin and try to correlate every nerve ending therein to every possible retinal cell in its eyes at every moment. Consequently, it might never learn some peculiar correlation between a particular patch of skin and a flash of light on some part of its retina, but presumably such correlations are not important to it in its natural environment.

Obvious physical dimensions along which to be perceptually selective include *spatial* and *temporal* selectivity.[2] The universe tends to display spatial locality: many causes are generally located nearby to their effects (for example, pushing an object requires one to be in contact with it). Further, many causes lead to an observable effect within a short time (letting go of an object in a gravity field causes it to start falling immediately, rather than a week later). Real creatures use these sorts of spatial and temporal locality all the time, often by using eyes that only have high resolution in a small part of their visual field, and only noticing correlations between events that take place reasonably close together in time. While it is certainly *possible* to conceive of an agent that tracks every single visual event in the sphere around it, all at the same time, and which can remember pairs of events separated by arbitrary amounts of time without knowing a priori that the events might be related, the computational burden in doing so is essentially unbounded.[3] The algorithm dis-

2. Another dimension of selectivity concerns the amount of preprocessing done to the input. For example, Drescher points out that, in a realistic world, correlating unprocessed retinal input is not very useful, because it does not map well onto aspects of the world that are good building blocks for inductive generalization [Drescher 94]. Such changes of level are not addressed in this research.

cussed in this research implements temporal selectivity as well as spatial selectivity to reduce the number of sensor data that the agent has to correlate with its internal structures (see Figure 1, on page 16). Note that the perceptual selectivity implemented is of a passive nature: the agent prunes its "bag of sensory bits," rather than changing the mapping of that bag of bits to the physical world by performing an action that changes the sensory data (such as changing its point of view). The latter would constitute active perceptual attention (e.g., [Aloimonos 93] among others).

2.1.3 Cognitive selectivity

Cognitive selectivity limits what internal structures are attended to at any given moment.

Notice that for any agent that learns many facts,[4] being cognitively selective is likely to be even more important than being perceptually selective, in the long run. The reasons for this are straightforward. First, consider the ratio of sensory to memory items. While the total number of possible sensory bits is limited, the number of internal structures may grow without bound.[5] This means that, were we to use a strategy which prunes all sensory information and all cognitive information each to some *constant fraction* of their original, unpruned case, we would cut the total computation required by some constant factor—but this factor would be much larger in the cognitive case, because the number of facts stored would likely far outnumber the number of sensory bits available.

3. Many algorithms for learning from experience employ an extreme form of temporal selectivity: the agent can only correlate events that are "one timestep" apart.
4. In this case, since we are discussing a causal model builder, these facts are correlations of actions and their results.
5. The assumption here is that every fact learned requires some internal structure to represent it. If the learning algorithm in use must examine prior facts to decide whether to invalidate the fact, create a new one, etc, then the computational effort of the algorithm will tend to increase as more facts are learned.

Second, consider a strategy in which a *constant number* of sensory bits or a *constant number* of remembered facts are attended to at any given time. This is analogous to the situation in which a real organism has hard performance limits along both perceptual and cognitive axes; no matter how many facts it knows, it can only keep a fixed number of them in working memory. In this case, as the internal structures grow, the organism can do its sensory-to-memory correlations in essentially constant time, rather than the aforementioned $O(n^2)$ time, though at a cost: as its knowledge grows, it is ignoring at any given time an increasingly large percentage of all the knowledge it has.

Compromise strategies which keep growth in the work required to perform these correlations within bounds (e.g., less than $O(n^2)$), yet not give in completely to utilizing ever-smaller fractions of current knowledge (e.g., more than $O(1)$) are possible.

For example, one can use properties of the world or characteristics of the sensor data to restrict the number of structures looked at (as is the case in the algorithms described in this thesis). Not all internal structures are equally relevant at any given instant. In particular, internal structures that refer neither to current nor expected future perceptual inputs are less likely to be useful than internal structures which do. This is the particular domain-independent, goal-independent, world-driven heuristic for cognitive selectivity which is employed in Chapter 3. One might argue that, in real creatures, evolution optimizes them to ignore those aspects of the environment which do not change; for example, there is little reason to perceive nor reason about the existence of air unless one is in an environment in which it is not ubiquitous.

Another way to compromise is to use the current *goal* to help select what facts are relevant; such *goal-driven pruning* is discussed in Chapter 4. Since generally only a small number of goals are likely to be relevant or applicable at any one time (often only one), this can help to keep the amount of correlation work in bounds.[6] Again, in real creatures

domain-dependent and goal-driven cognitive selectivity play a large role too. For example, the subset of internal structures that are considered at some instant is not only determined by what the agent senses and what it expects to sense next, but also by what it is "aiming" to sense or not sense (i.e., the desired goals).

2.2 The testbed microworlds

In order to evaluate the effects of adding focus of attention to a system that can learn a world model, we need something which learns (the *agent*) and some world model for it to learn in (the *domain* or *environment* of the agent). The two agents chosen here were both software agents, operating in a a *microworld* consisting of a simulated environment. The combination of a particular agent and its associated environment is called a *scenario*.

The sections immediately following describe the two scenarios used here. Later, in Section 3.2.1, on page 40, we describe what the learning system does with the sensory information it receives from either environment.

In both scenarios, the world may be unpredictable; actions are allowed to have no perceivable effect for any of several reasons, including incomplete sensor information (in neither scenario does the agent have an all-encompassing sensor view of the world), and external motions or actors in the world that cannot be controlled by the agent.

In general, the learning system is connected "at arm's length" to either of the two microworlds, and can issue only one of a small number of commands at each timestep of the simulation. It gets back a collection of sensory bits describing what the simulated sensors are perceiving, and does not have any other access to the internal state of the micro-

6. Another way to compromise might be to investigate much more of memory when other demands on the agent's time are minimal, essentially doing as much extra work as possible when not otherwise occupied with immediate concerns or goals which must be completed under tight deadlines.

world. Actions are taken at random in the case of goal-independent learning, but are informed from the goal system when performing goal-dependent learning.

Appendix A describes the overall system architecture used in this research, and shows a schematic of how the learning system, the action selection system, and the goal system interact. Figure 20, on page 132, describes what pieces of the system communicate with which other pieces, and Figure 21, on page 134, shows details of when the learning system and the microworld are allowed to exchange information. Appendix A also provides some implementation details specific to this particular system, which might be of use in reproducing it.

2.2.1 The infant/eyehand scenario

The most extensively studied scenario, both in prior work with the particular learning system employed [Drescher 91], and in this research, concerns a simulated infant in a simple, mostly (but not completely) static microworld.

Drescher's original system is concerned with Piagetian modeling, so his microworld is oriented towards the world as perceived by a very young infant (younger than eight months old, e.g., before early fourth Piagetian stage). The simplified microworld, shown in Figure 2, on page 23, consists of a simulated, two-dimensional "universe" of 49 grid squares (7 by 7). Each square can be either empty or contain some object. Superimposed upon this universe is a crudely simulated eye which can see a square region 5 grid squares on a side, and which can be moved around within the limits of the simulated universe. This eye has a fovea of a few squares in the center, which can see additional details in objects (these extra details can be used to differentiate objects enough to determine their identities). The universe also includes a hand which occupies a grid square, and can bump into

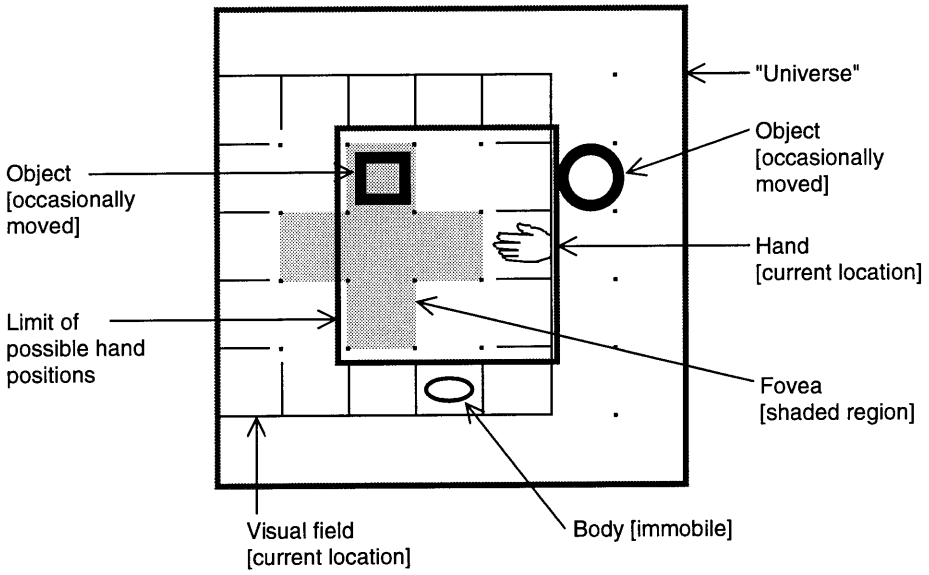

Figure 2: The eyehand/infant domain microworld

and grasp objects. (The infant's arm is not represented; just the hand.) An immobile body occupies another grid square.

The infant has 10 possible actions that it can take at any given timestep, consisting of moving the hand or eye forward, backward, left, or right (8 actions in all), and of opening or closing the hand. It takes one of these actions at every timestep.

The possible sensory inputs consist of all bits arriving from the eye, proprioceptive inputs from eye and hand (which indicate where, relative to the body, the eye is pointing or the hand is reaching), tactile inputs from the hand and body, and taste inputs from the mouth (if an object was in contact with it).[7] The eye reports only whether an object (not *which* object, only the presence of one) is in a grid square or not, except in its central fovea, where it reports many additional bits.[8]

7. The hand gets four one-bit details when it is in contact with an object on its left side, which can be used to differentiate objects by touch. It also gets one bit per side indicating whether an object is in contact with it on that side. Similarly, the body gets one bit apiece to indicate contact on any of its four sides; if an object is in contact on the front side of the body (the mouth), then four bits of taste information are also available. Finally, there is one bit each representing whether the hand is currently closed and whether it is grasping an object.

The simulated infant does *not* have a panoramic view of all 49 squares of the universe at once; at any given instant, it only knows about what the eye can see, what the hand is touching, or what the mouth is tasting, combined with proprioceptive inputs for eye and hand position. In particular, certain senses, viewed unimodally, are subject to perceptual aliasing, in which two distinct situations in the environment appear identical to the sensory system of the agent [Whitehead and Ballard 90]. For example, if a schema mentions only a particular bit in the visual field, without also referring to the visual proprioceptive inputs (which determine where the eye is pointing), then that schema may be subject to such aliasing—several different situations have been collapsed into the same representation, as far as the agent is concerned. Similarly, any schema mentioning any visual-field sensory item that is not in the fovea may alias different objects, since the non-foveal visual field reports only the presence or absence of an object in each position, rather than the exact identity of the object in question.

Typical knowledge that is learned about the world includes correlations between motion of the hand and motions of the image of the hand in the visual field; motions of the eye and motions of all objects in the visual field; correspondences between proprioceptive and haptic or visual information; whether or not an object will be graspable depending on whether or not it is felt to be in contact with the hand, and many more ([Drescher 91] and [Ramstad 92] report at length the many unimodal and multimodal facts about the micro-world learned by this system).

8. Each of the five foveal squares provides 25 bits of detail information. Each different object sets a different combination of these 25 bits; hence, objects may be differentiated visually when they are in a foveal square, because different combinations of bits are perceived. Note that each foveal square covers one coarse visual square, and all motion is quantized by these coarse visual squares: this means that the details corresponding to some object will never be half on one foveal square and half on another. To put this another way, if an object is somewhere in the fovea, there are only five different positions (corresponding to the five different foveal squares) that any one object can be in. Objects are never rotated.

Typical strategic goals for the infant in this microworld include centering an object in the visual field, moving the hand into proximity to an object in order to grasp it, and so forth. Clearly, such goals cannot be accomplished in reasonable time without having learned about the effects of the infant's actions.[9]

2.2.2 The Hamsterdam scenario

The second agent and environment used in this research were based on the Hamsterdam system [Blumberg 94]. This system's primary use is for investigating ethological models of action-selection [Blumberg 94], and it is also used as a major component of the "magic mirror" virtual environment created for the ALIVE project [Darrell 94] [Maes 93].

The Hamsterdam system consists of a three-dimensional world in which simulated hamsters and predators may interact. The world also includes a floor, walls, food and water sources, and so forth. A second instantiation of the ALIVE project included a humanoid puppet, which ran under control of a simple finite-state machine, rather than the more complicated, ethologically-based controllers used for the hamsters and the predators. The entire microworld ran in real time, and was rendered as it ran using SGI Inventor on a Silicon Graphics workstation. Figure 3, on page 26, shows a typical scene from Hamsterdam, in which a hamster is in the lower left, and a predator is in the upper right. (The predator is currently unable to escape its box of walls, but a human outside of the simulation has the ability to slide the wall aside and enable the predator to reach the hamster.) Figure 4, immediately below Figure 3, shows the puppet, pictured standing alone in the world.

9. The phrase "reasonable time" is important: since the world is relatively small and the action set constrained, it might be possible to accomplish certain strategic goals by taking *random* actions and eventually reaching the goal by luck. However, as demonstrated in Section 4.4.3, on page 91, the average number of actions employed to reach a typical strategic goal are at least an order of magnitude shorter when the agent has learned the consequences of its actions than when the agent is not allowed to learn.

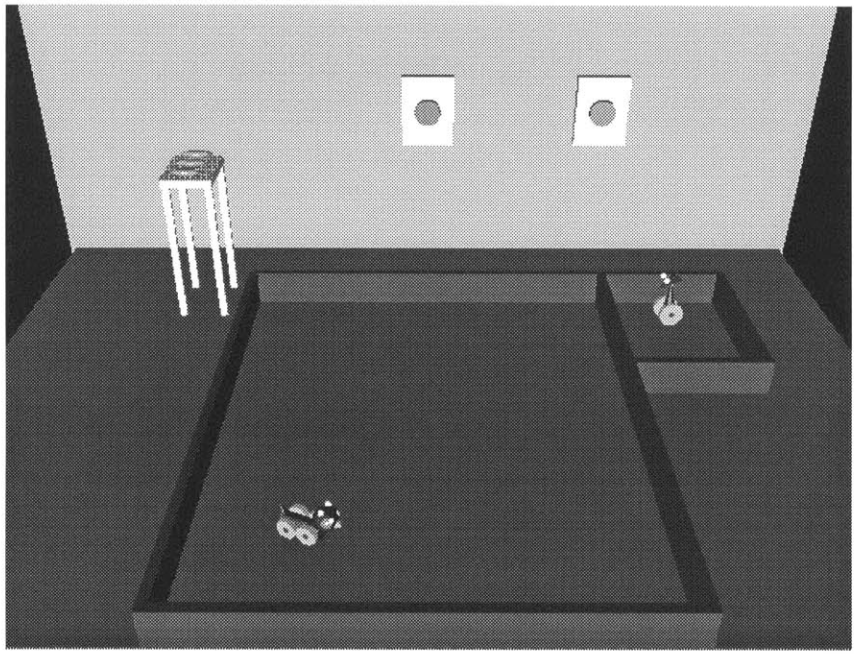

Figure 3: Hamsterdam with a hamster and a predator

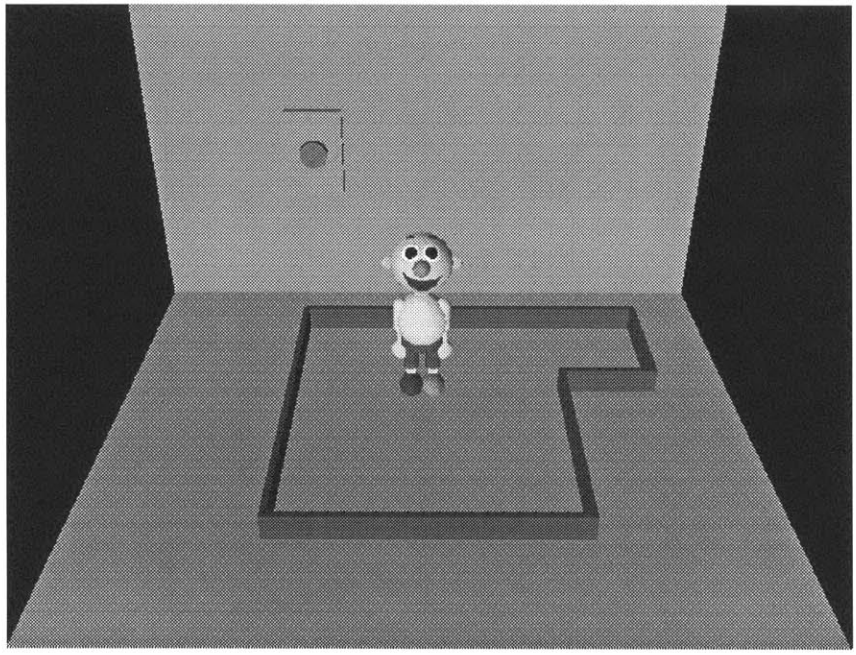

Figure 4: Hamsterdam with a puppet

The animals in the original Hamsterdam system have a sensory system resembling that of a robot sonar system. They shoot out 15 simulated rays in a horizontal fan at floor level, subtending 180 degrees total, and get back sonar-like echoes which indicate, along each ray, how close any given object is. The echo actually reveals the identity and type of any given object, rather than simply reporting that "something" is there. A typical representation of such a sensor fan appears in Figure 5, below, as rendered by Hamsterdam, in which

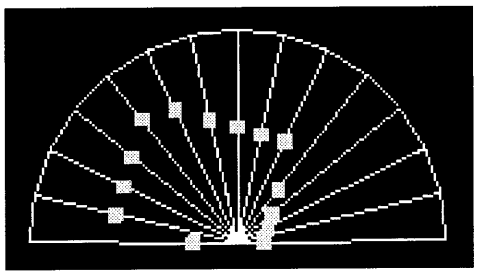

Figure 5: Hamsterdam sensor fan

the white squares at various distances along the radials are the sensor echoes of (in this case) part of a cul-de-sac where several walls meet. This sensory system, unlike that in the infant/eyehand scenario, is not particularly multimodal. The information returned for an object consists of the range r and angle θ at which the object is sighted: this is essentially purely visual. No proprioceptive information is available. The qualities reflected by taste and texture in the infant/eyehand scenario are reflected most closely by the tags returned by the sensor system, which indicate object identity and type.

Since the underlying learning system used in this research (e.g., the *schema system* [Drescher 91], without the composite-action system) requires all sensory information to be reduced to individual boolean predicates, rather than, e.g., numbers or ranges (see Section 3.2.1, on page 40), the results of the sensory fan must be discretized. This is accomplished as follows.

First, positions in the polar coordinate system defined by r and θ are quantized. The original, continuous value of r is reduced to one of five ranges, compressing r values to only five distinct quantities. These ranges are, in fact, scaled logarithmically, so that ranges which are farther from the agent cover more distance in the unscaled, original space; this provides more detailed range information for objects which are nearby without requiring a large number of additional ranges.

If an object is at some particular combination of r and θ, the sensory item (a single bit) corresponding to that combination is turned on. Since there are 15 radials and 5 ranges, this means that there are 75 sensory items devoted to *r/theta* information. Note that these sensory items cannot be used to differentiate one object from another, but merely to indicate that an object occupies that particular position. In this respect, they are similar to the coarse visual field items in the infant/eyehand scenario.

In addition to these inputs, the sensor fan is also "foveated" to yield higher-quality information at short ranges in the center of the fan. The central three radials, for the nearest four ranges, also return one bit apiece to indicate whether an object at that position is one of: a hamster, a predator, a wall, food, or water. The cross product of these 3 by 4 by 5 possibilities yields another 60 sensory items.

The discretized sensory system is illustrated in the schematic below. The foveation is

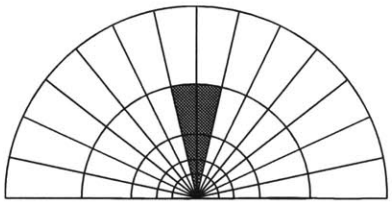

Figure 6: Discretization and foveation of the Hamsterdam sensor fan

shown by the gray region. Note that the logarithmic scaling of r is actually more pronounced than shown in this diagram

This sensory system has all of the possible problems with sensory aliasing described in Section 2.2.1, on page 22, in the description of the infant/eyehand scenario, and more. In particular, the infant/eyehand scenario at least has proprioceptive inputs from its eye, so it could (in theory) be able to build a complete world picture by taking into account the current portion of the world encompassed by its eye (as revealed by proprioceptive inputs), combined with current inputs from the eye itself. In the Hamsterdam scenario, even that level of information is unavailable, since the actors in the world are free to roam about it, and information about their current position or orientation is not available from the sensory system (and would have to be inferred either from available sensory information, dead reckoning from a known landmark, or something similar). If the infant/eyehand scenario did not make proprioceptive sensory items available from the hand and eye, it would resemble this aspect of the Hamsterdam scenario.

In the ALIVE system [Maes 93], the puppet never "shared the stage" with the hamsters and the predators; instead, users of the systems could switch between these two worlds. As modified for this research, the puppet and the animals are both allowed to share the same world.

Further, the puppet in ALIVE did not have an ethologically-based action controller; instead, inputs from the visual tracking system which tracked human participants drove a simple finite-state machine which in turn commanded various movements of the puppet. As modified here, the puppet's actions are controlled by the learning system, and it has had the hamster sensory system "grafted on" to serve as input sensory items to the learning system.

The learning system may only control the puppet, and not any of the other animals or, e.g., the position of the walls. The allowable motions of the puppet include walking for-

ward a step, turning in either direction,[10] standing up from a sitting position, and sitting down again.

As in the infant/eyehand scenario, learning the consequences of actions includes learning the correlation between actions such as rotating or walking and the observed movement of objects in the sensor fan. Goals include rotating until an object is centered along a foveal ray, walking until an object is at minimum range, and so forth.

Because the Hamsterdam scenario is significantly more active than the infant/eyehand scenario (e.g., hamsters or predators may be rolling around all the time, causing many changes to the environment regardless of what the agent is doing), the learning system has an explicit representation of a *null action* in this scenario, used to help model the *result of not doing anything*. See Appendix A for further details about null actions.

10. The puppet turns by an amount which matches the angular offset of a pair of rays in the sensor fan, e.g., about 12.857 degrees. This was chosen to conveniently match the action to the sensory system, in the way that moving the eye or hand in the eyehand/infant scenario causes all the other sensory inputs (e.g., proprioceptive, visual) to slide one position in some direction.

Chapter 3: Goal-independent Learning

This research started with an existing learning algorithm [Drescher 91], and added a focus of attention mechanism, as described below, to make learning faster (requiring less computation per timestep). While Drescher's original work does include a concept similar to both tactical and strategic goals, his system does not exploit goals to guide the learning process. Further, it has no perceptual selectivity and assumes that every sensory bit might be useful all the time.[1] This approach leads to a theoretically "pure" result for the issues which Drescher was investigating, but one which is difficult to use in a real application, and which seems somewhat implausible in describing how real organisms learn.

The following discussion is an overview of the operation of the schema system; for a much more complete and detailed explanation of its intricacies, see [Drescher 91] and [Ramstad 92].

3.1 Model of the learning system

This section presents a very brief overview of the schema system [Drescher 91] to aid in understanding the focused and unfocused learning algorithms presented later. The schema system is designed to be used in building causal models of the results of actions.

1. With one very small exception, as follows. The last action taken is itself represented in the bits given to the learning algorithm (since any new schema created to represent the results of the action just taken will be a schema mentioning that action, and no other). Since *only* the last action taken is so represented to the learning algorithm, *only* the last action taken is attended to when attempting to correlate actions with their results. All other sensory inputs (e.g., proprioceptive, visual, etc) are attended to whether or not they have changed recently.

Hence, it is designed to be hooked up to some agent which is situated in an environment, and which has some sensory inputs and motor outputs.

3.1.1 The sensory system: Items

Every sensory item is represented in the simulation as a single bit. In Drescher's original algorithm, there is no grouping of these bits in any particular way (e.g., as a retinotopic map, or into particular modalities); the learning algorithm sees only an undifferentiated "bag of bits."

This means that values in the world which are not simple binary bits must somehow be turned into individual bits. Ranges, for example, are most usefully represented as a set of bits, any one of which would be asserted when the value is within some particular interval of the range.[2]

3.1.2 The motor system: Actions

The schema system assumes a known, fixed number of possible actions that the agent may take. It takes one of these actions at every timestep of the simulation, and observes which sensory items have changed after the action.

In Drescher's original implementation, the system spent most of its time (a fixed 90%) taking random actions and observing the results. The remaining 10% of its time was spent taking actions which had led to some reliable outcome before, to see if actions could be chained.[3] The work reflected in this chapter instead represents a totally random action selection strategy: the agent always picks its next action at random. In Chapter 4, in the

2. There is no particular reason why such range values could not be represented instead as, e.g., a binary number. However, such a representation, though more compact in terms of the number of sensory bits, would be somewhat slower to learn, since a schema (see Section 3.1.3) which made accurate predictions about the value might have to have a conjunctive context or result (see Section 3.1.3.2.6) which completely specified all n bits of the number. Generating such conjunctions would be quite timeconsuming.
3. See the composite action mechanism of [Drescher 91].

context of goal-directed focus of attention, we also investigate less random and more goal-oriented action selection strategies.

3.1.3 The knowledge base: Schemas

3.1.3.1 What is a schema?

The "facts" learned by this system are called *schemas*. They consist of a triple of the *context* (the initial state of the world, as perceived by the sensory system), the *action* taken on this iteration, and the *result* (the subsequent perceived state), which maps actions taken in a particular configuration of sensory inputs into the new sensory inputs resulting from that action. In the infant/eyehand scenario, a typical schema might therefore be, "If my eye's proprioceptive feedback indicates that it is in position (2,3) [*context*], and I move my eye one unit to the right [*action*], then my eye's proprioceptive feedback indicates that it is in position (3,3) [*result*]." A simple textual way of representing this would be VP23/EYER/VP33, as described in [Drescher 91] and [Ramstad 92].

Another typical schema might be, "If my hand's proprioceptive feedback indicates that it is in position (3,4) [*context*], and I move it one unit back [*action*], I will feel a taste at my mouth and on my hand [*result*]," or HP34/HANDB/TASTE&TACT.[4] Notice that this latter schema is *multimodal* in that it relates a proprioceptive to taste and tactile senses; the learning mechanism and its microworld build many multimodal schemas, relating touch to vision, vision to proprioception, taste to touch, graspability to the presence of an object near the hand, and so forth. It also creates *unimodal* schemas of the form illustrated in the first schema above, which in this case relates proprioception to proprioception.

Since behavior of the world may be nondeterministic (e.g., actions may sometimes fail, or sensory inputs may change in manners uncorrelated with the actions being taken), each

4. This is because the hand will move from immediately in front of the mouth to in contact with the mouth. We assume that the infant's mouth is always open, which seems empirically reasonable for most infants.

schema also records statistical information which is used to determine whether the schema accurately reflects a regularity in the operation of the world, or whether an initial "guess" at the behavior of the world later turned out to be a coincidence. (For details about how this actually works, see the next section.) This information is recorded in a schema's *extended context* and an *extended result,* which keep these statistics for every item not yet present in the context or result.

3.1.3.2 How are schemas created? The marginal attribution mechanism

A schema is deemed to be *reliable* if its predictions of **(context, action, result)** are accurate more than a certain percentage of the time. If we already have a reliable schema, and adding some additional sensory item to the items already expressed in either its context or its result makes a schema which appears that it, too, might be reliable, we *spin off* a new schema expressing this new conjunction.[5] *Spinoff schemas* satisfy several other constraints, such as not ever duplicating an existing schema, and may themselves serve to be the basis for additional spinoffs later.

The description below, until the end of this section (Section 3.1.3.2), is quoted with a few modifications from [Ramstad 92]; that description, in turn, was a summarization of [Drescher 91]. See the latter in particular for more depth on the subtleties of the schema system and marginal attribution in particular; this description, being a summary, glosses over particular refinements which prevent certain combinatorial explosions.

5. For example, consider the schema VF33/EYER/FOVF22, which states that, if a particular coarse visual field bit (the one at 3,3) is on, and the agent moves the eye right, it will see a particular fine foveal visual bit turn on (the one at 2,2). If this schema is sufficiently reliable, and it also seems from experience that VF32 is usually off just before we take this action in this case, the schema system might spin off the schema ¬VF32&VF33/EYER/FOVF22 from the original schema. (Note that the notation often used in, e.g., [Ramstad 92] would say -VF32 instead of ¬VF32; we use both forms interchangeably here.) While the *original* schema must be deemed reliable to be considered for a spinoff (to prevent combinatorial explosion), the *new* one is not yet known to be reliable, and will itself be considered a spinoff candidate only if it, too, is later shown to be reliable.

3.1.3.2.1 Bare schemas

Schemas are bootstrapped from an initial set of *bare* schemas, which have neither context nor result and hence do not predict anything. There is one bare schema for each action that can be taken.

As the simulation is run, a technique known as *marginal attribution* is used to discover statistically important context and result information. This information is then used to fine-tune existing schemas by creating modified versions of them. Marginal attribution succeeds in greatly reducing the combinatorial problem of discovering reliable schemas from an extremely large search space without prior knowledge of the problem domain.

3.1.3.2.2 Result spinoffs

Many different results may occur from the execution of a given action. For every *bare* schema, the schema mechanism tries to find result transitions which occur more often with a particular action than without it. For example, a hand ends up closed and grasping an object much more often when the *grasp* action is taken than with any other action. Results discovered in this fashion are eligible to be included in a *result spinoff*—a new schema identical to its parent, but with the relevant result item included. The marginal attribution process can only create result spinoffs from bare schemas.

Specifically, each bare schema has an *extended result*—a structure for holding result correlation information. The extended result for each schema keeps correlation information for each item (primitive or synthetic). The positive-transition correlation is the ratio of the number of occurrences of the item turning on when the schema's action has been taken to the number of occurrences of the item turning on when the schema's action has not been taken. Similarly, the negative-transition correlation is the ratio of the number of occurrences of the item turning off when the schema's action has been taken to the number of occurrences of the item turning off when the schema's action has not been taken. Note that

an item is considered to have turned on precisely when the item was off prior to the action and on after the action was performed, and similarly for turning off. The correlation statistics are continuously updated by the schema mechanism and weighted towards more recent data. When one of these schemas has a sufficiently high correlation with a particular item, the schema mechanism creates an appropriate result spinoff—a schema with the item positively included in the result if the positive-transition correlation is high, or a schema with the item negatively included in the result if the negative-transition correlation is high. These simple statistics are very good at discovering arbitrarily rare results of actions, especially when the statistics of the non-activated schemas are only updated for *unexplained* transitions. A transition is considered explained if the item in question was included in the result for an activated schema with high reliability (above an arbitrary threshold) and it did, in fact, end up in the predicted state.

3.1.3.2.3 Context spinoffs

For schemas which have non-negligible results, the marginal attribution mechanism attempts to discover conditions under which the schema obtains its result more reliably. To extend the example cited above, a hand ends up closed and grasping something much more often an object can be felt touching the correct part of the hand before the grasp action is executed. This information is used to create *context spinoffs*—duplicates of the parent schema, but with a new item added to its context.

Schemas with non-empty results have an *extended context*. For each item, this structure keeps a ratio of the number of occurrences of the schema succeeding (i.e., its result obtaining) when activated with the item on the number of occurrences of the schema succeeding when activated with the item off. If the state of a particular item before activation of a given schema does not affect its probability of success, this ratio will stay close to unity. However, if having the item on increases the probability of success, the ratio will increase over

time. Similarly, if having the item off increases the probability of success, the ratio will decrease. If one of these schemas has a significantly high or low ratio for a particular item, the schema mechanism creates the appropriate *context spinoff*—a schema with the item positively included in the context if the ratio is high, or one with the item negatively included if the ratio is low.

There is an embellishment to the process of identifying context spinoffs. When a context spinoff occurs, the parent schema resets all correlation data in its extended context, and keeps an indication of which item (positively or negatively included) was added to its spinoff child. In the future, when updating the extended context data for the parent schema, if that item is on (if positively included in the spinoff) or off (if negatively included in the spinoff), the trial is ignored and the extended context data is not modified. This embellishment means that the parent schema has correlation data only for those trials where there is no more specific child schema, and it encourages the development of spinoff schemas from more specific schemas rather than general schemas.

Redundancy is also reduced by a further embellishment. If, at a particular moment in time, a schema has multiple candidates for a context spinoff, the item which is on least frequently is the one chosen for a context spinoff. The system keeps a *generality* statistic for each item which is merely its rate of being on rather than off—it is this statistic which is used when deciding between multiple spinoff possibilities. This embellishment discourages the development of unnecessary conjunctions when a single specific item suffices.

3.1.3.2.4 Conjunctive contexts and results

The context can be iteratively modified through a series of context spinoffs to include more and more conjuncts in the context. For a variety of reasons, but primarily to avoid combinatorial explosion, a similar process for result conjunctions is undesirable. The marginal attribution process therefore requires that result spinoffs occur only from bare sche-

mas, and only one relevant detail can be detected and used as the result for the spinoff schema. However, conjunctive results are necessary if the schemas should be able to *chain* to a schema with a conjunctive context. (See Chapter 4 for much more about chaining schemas.) This problem is solved by adding a slot to the extended result of each bare schema for each of the conjunctions of items which appear as the context of a highly reliable schema. Statistics are kept for these in the same fashion as those kept for single items, and if one of these conjunctions is often turned on as the result of taking a given action, a result spinoff occurs which includes the entire conjunction in the result. Effectively, this process is able to produce schemas with conjunctive results precisely when such schemas are necessary for chaining.

3.1.3.2.5 Summary of marginal attribution

Schemas created by the marginal attribution process are designed to either encapsulate some newly discovered piece of knowledge about causality in the microworld (result spinoff), or to improve upon the reliability of a previous schema (context spinoff). By continuously creating new versions of previous schemas, the system iteratively improves both the reliability and the scope of its knowledge base. It is interesting to note that, once created, a schema is never removed from the system. Rather, it may be supplanted by one or more spinoff schemas which are more *useful* due to higher reliability and greater specificity.

3.1.3.2.6 Synthetic items

The schema system also defines many other concepts; we mention here briefly one other, namely *synthetic items*, that will become more important in Section 4.4.2, on page 87. This section can safely be skipped, at least until then, without loss of continuity.

There are certain concepts that primitive items are unable to express, for example, that a particular object is present at a particular location while the glance orientation is such that

the object is out of view. The schema mechanism contains a facility for building *synthetic items*—items which, when on, indicate that a particular unreliable schema, if activated, would succeed. Suppose a schema saying /[MOVE GLANCE ORIENTATION TO VP01]/VOVF02 (note that this schema has no context) is very reliable if some object in the microworld is in the correct position. However, this object spontaneously moves around between a few different positions and so, on average, is only in the correct position some of the time. Notably, this schema, if activated and successful, will continue to be very reliable for some period of time (equal to the duration that the object remains in that position), even though on average it is normally not very reliable. To discover such situations, the schema mechanism keeps a *local consistency* statistic which indicates how often the schema succeeds when its last activation was successful. If a schema is unreliable but highly locally consistent, the mechanism constructs a synthetic item—an item which, when on, indicates that the schema (its *host schema*), if activated, would succeed. Effectively, such an item, when on, predicts what the result of activating the host schema would be. For a variety of reasons (see [Drescher 91]), synthetic items are fundamentally very different from primitive items and express concepts which are inexpressible through any conventional combination of primitive items.

Primitive items get their state directly from the microworld. On the other hand, the schema mechanism itself must maintain and update the state of all synthetic items. (The rules for how this update is accomplished are complicated and not explained here.) The use of synthetic items effectively allows the schema mechanism to invent new concepts—concepts which are not expressed well by the microworld or cannot be expressed by conjunctions of boolean values at all.

3.2 Improving learning via focus

If we are to talk about improving the computational effort of learning, we must first be clear about what happens during learning, and how to describe how much work is performed for various modifications of the basic learning algorithm.

3.2.1 The work of learning

Given the action model learning algorithm described above, at each clock tick, we must do two things:

- First, we must update various statistics reflecting what just happened; this is the "perception" part of the learning algorithm. A focus mechanism dictates which sensory items will be attended to[6] ($Stat_i$), for which schema numbers ($Stat_s$).
- Second, we must decide whether to spin off a new schema; this is the "learning" part of the algorithm, and here the focus mechanism dictates which item numbers to check for reliability ($Spin_i$) for which schema numbers ($Spin_s$).

Thus, our choice of these four sets of numbers determines which sensory items and schemas are used in either updating our perception of the world, or deciding when a correlation has been learned. $Stat_i$ and $Stat_s$ determine the perceptual selectivity, while $Spin_i$ and $Spin_s$ determine the cognitive selectivity of the agent, as shown in the table below)

	Statistics	**Spinoffs**
Which items?	$Stat_i$	$Spin_i$
Which schemas?	$Stat_s$	$Spin_s$

6. In other words, will have their associated statistics reflecting frequency of occurrence updated in all schemas which mention them.

The algorithm does the vast majority of its work in two $O(n^2)$ loops (one loop for updating statistics, reflecting what is currently perceived, and one loop for deciding whether to spin off new schemas, reflecting learning from that perception). The number of items and the number of schemas selected at any given clock tick determines the amount of work done by the learning algorithm in that tick. Thus, if $Work_{Stat}$ is the work done during the statistical-update part of the algorithm (e.g., perceiving the world) of any one clock tick, and $Work_{Spin}$ is similarly the amount of work done in deciding which spinoff schemas to create, then the work done during either one is the product of the number of items attended to and the number of schemas attended to, or:[7]

$$Work_{Stat} = \|Stat_i\| \bullet \|Stat_s\|$$

$$Work_{Spin} = \|Spin_i\| \bullet \|Spin_s\|$$

This means that the total work per step (clock tick) is simply the sum of these individual pieces, and that the total work over some particular number of steps is simply the sum of the work during the individual steps:[8]

$$Work_{Step} = Work_{Stat} + Work_{Spin}$$

$$Work_{Total} = \sum_{steps} Work_{Step}$$

Thus, the behavior of $Work_{Step}$ over time tells us how well the algorithm will do at keeping up with the real work, e.g., how much it slows down as the number of iterations (which is proportional, though not in a particularly simple way, to the number of schemas) increases.

7. Where $\|x\|$ denotes the cardinality of the set x.

8. There is no particular end to this series of learning steps. In this research, the number of steps taken was limited by the amount of time available to learn, or by the approximate number of schemas that were desired for a run, and so forth. In a real agent, we might turn off the learning system once the agent has shown that it is competent (see Chapter 4) to perform some set of tasks—or not. After all, if we were ever to turn off the learning system, the agent would fail to change its internal world model even if the external world were to change.

The discussion above looked at the work per step, e.g., per clock tick of the simulation. Another way of evaluating the utility of the various algorithms is to examine the amount of work performed per schema learned (either reliable schemas or all schemas; the former being perhaps the more useful metric):

$$Work_{Step}SchemaRel = \frac{Work_{Step}}{SchemaRel_{Step}}$$

which is simply the amount of work performed during some steps ($Work_{Step}$) divided by the number of reliable schemas generated by that work ($SchemaRel_{Step}$).[9] A similar definition for total schemas over total work is straightforward:[10]

$$Work_{Step}SchemaTotal = \frac{Work_{Step}}{SchemaTotal_{Step}}$$

An algorithm which determines these choices is thus described by the pairs $<<Stat_i,Stat_s>,<Spin_i,Spin_s>>$; we will call each element a *selector.*

A little more terminology will enable us to discuss the actual selectors used. $Schema_{max}$ is the number of schemas currently learned. $Item_{max}$ is the number of sensory items. $Item_n$ is the value of sensory bit n, and $Item_{n,t}$ is the value of that item at some time t. $SchemaDep_sOn_i$ denotes some schema s which is dependent upon (e.g., references in its context or result) some item i.

9. Note a peculiar detail here. It is possible for a schema that was formerly thought to be reliable to be later decided to be unreliable. This could happen if the world has changed in the meantime, or if some not-very-correct correlation happened often enough to push the schema over the arbitrary threshold from not being considered reliable to being considered reliable, and then later data pushed the schema's reliability back down. Hence, it is possible for the number of reliable schemas to *decrease* during a single step, and this is not an altogether infrequent occurrence. This means that, while the *average* of $Work_{Step}SchemaRel$ is positive, the instantaneous value might be negative if *Step* is a single step or a small number of steps.

10. Since the *total* number of schemas (as opposed to *reliable* schemas) can never decrease, this number must always be nonnegative.

3.2.2 The basic learning algorithm

In Drescher's basic algorithm, every possible sensory bit *before* an action taken by the "infant" was correlated with every possible sensory bit *after* the action, for *every* schema that has been created so far. In other words, $Stat_i$ and $Spin_i$ use the selector **all item numbers**, or **AIN**:

$Stat_i = Spin_i = \text{AIN}$, where

$\text{AIN} = \{n | 0 \leq n \leq Item_{max}\}$

and $Stat_s$ and $Spin_s$ use the selector **all schema numbers**, or **ASN**:

$Spin_i = Spin_i = \text{ASN}$, where

$\text{ASN} = \{n | 0 \leq n \leq Schema_{max}\}$

This means that the basic algorithm does a tremendous amount of work in the two n **x** m inner loops, where n is the size of the set of items in use, $\|\text{AIN}\|$, and similarly m is the size of the set of schemas in use, $\|\text{ASN}\|$. Hence,

$Work_{Stat} = Work_{Spin} = \|\text{AIN}\| \bullet \|\text{ASN}\|$

This algorithm can eventually learn a large number of facts about the world in this way, but it runs slowly, and becomes increasingly slow as the number of known facts (e.g., schemas) increases. Furthermore, if we were to increase the number of sensory bits available (e.g., by putting a higher-resolution camera in an agent using this technique), the work involved would increase in direct proportion to the number of added sensory bits, even if none of these bits ever changed.

3.2.3 The focused algorithm

Various pruning techniques help a great deal over the basic approach. The most successful of the approaches examined, which we shall call the *focused* algorithm, takes the following tack (why these particular parameters were chosen is explained at the end of this section):

- *Perceptual selectivity.* When updating statistics, only consider sensory items which have changed very recently (last two clock ticks) and only in schemas which make predictions about those items.
- *Cognitive selectivity.* When deciding whether to spin off a schema (make a new fact), only consider sensory items which have changed in the last clock tick, and only consider schemas which have had their statistics changed in the last clock tick (such schemas can only have had their statistics changed if they themselves made predictions involving sensory items which themselves have changed).

Put more precisely, the items used were as follows. $Stat_i$ used the **all changed items in history**, or **CINIH** selector (where the word "history" refers to a timeline of prior events, of some chosen length or *horizon*, and in this case of length 2):

$Stat_i$ = CINIH_H, where

$$\mathrm{CINIH}_H = \mathrm{AIN} \cap \{\exists (0 < T \leq H) \mid Item_{n,t} \neq Item_{n,t-T}\}$$

while $Spin_i$ used a specialization of this, in which the history is only the very last event, which we shall call the **changed item numbers**, or **CIN**, selector for compactness:

$Spin_i$ = CIN, where

$$\mathrm{CIN} = \mathrm{AIN} \cap \{Item_{n,t} \neq Item_{n,t-1}\} = \mathrm{CINIH}_1$$

Similarly, the schemas used were as follows. Consider the set **all bare schemas**, or **ABS:**[11]

$$\text{ABS} = \text{ASN} \cap \{\forall (0 \le i \le Item_{max}) \mid \neg SchemaDep_n On_i\}$$

To define $Stat_s$, we add to these **schemas dependent upon changed items**, to get:

$Stat_i$ = ABSPSDUCI$_H$, where

$$\text{ABSPSDUCI} = \text{ABS} \cup \{SchemaDep_n On_{i \in \text{CIN}}\}$$

This selector is a special case of the more general one (which uses an arbitrary-length history), in that it uses a history of length 1. The general case, of course, is:

$$\text{ABSPSDUCIIH}_H = \text{ABS} \cup \{SchemaDep_n On_{i \in \text{CINIH}_H}\}$$

Finally, $Spin_s$ is defined as schemas with recently updated statistics, or

$Spin_i$ = SWRUS, where

$$\text{SWRUS} = \text{ASN} \cap \{SchemaDep_n On_i \mid (i \in \text{CIN})\}$$

Adding these changes amounts to adding some simple lookup tables to the basic algorithm that track which items were updated in the last clock tick, and, for each item, which schemas refer to it in their contexts or results. These tables are then used to determine which sensory items or schemas will be participants in the statistical update or spinoffs. Keeping the tables updated requires a negligible overhead on the basic algorithm.[12]

3.2.3.1 How has focus changed the computational effort?

The computational effort in the two n x m major loops is reduced by these selectors as follows:

11. Recall, from Section 3.1.3.2, on page 34, that bare schemas make no predictions about anything, and that there is one of these per action at the start of any run.

$$Work_{Stat} = \|\text{CINIH}_2\| \bullet \|\text{ABSPSDUCI}\|$$

$$Work_{Spin} = \|\text{CIN}\| \bullet \|\text{SWRUS}\|$$

In any run which generates more than a trivial number of schemas or has more than a handful of sensory bits, this is a dramatic reduction in the complexity, as shown in Figure 8, on page 55. Another way to look at it is as follows:

- The *unfocused* algorithm allows the work of learning to grow as the full cross-product of the *total* number of sensory bits (items) and the *total* number of predictions we make about the world (schemas).
- In the *focused* algorithm, the work of learning instead grows as the product of the amount of *change* in the world times the number of predictions we make about *those items which changed.*

If the world were such that *every* item changed at *every* step, and we had (somehow) managed to make a prediction about *every* item in *every* schema, then these two algorithms would behave identically. However, this does not describe very many plausible worlds in which we might want an agent to do learning, nor is it plausible that every prediction the agent may want to make about the world needs to mention every possible sensory bit the agent can perceive.

12. *Sensory*: Updating the list of items which have changed recently (where *recently* is defined by the horizon in use) runs in time linear with the number of items.

Cognitive: Each time we spin off a schema, we must update the table that maps items to schemas which depend upon them, in order to properly reflect the dependence of the schema on the items. This update runs in time linear with the number of items mentioned by the schema, and this number is very small anyway (less than half a dozen or so in the runs described).

Result 1: The work per spinoff is linear, which means it is negligible compared to the rest of the algorithm, which runs overall in approximately $O(n^2)$ time.

Result 2: We must perform the cognitive step above for each spinoff. Spinoffs become somewhat more frequent per clock tick as the number of clock ticks increases (e.g., if there are more schemas in the knowledge base, the number of new, spinoff schemas we are likely to create is higher). Thus, there is a slow growth in the overall work per clock tick to keep these tables updated, which grows per clock tick as the number of spinoffs per clock tick grows.

The problem of learning here is still $O(n^2)$, of course. We are still correlating *some* sensory bits with *some* predictions. However, the magnitude of this correlation has been decreased by an amount reflecting the behavior of the world and the characteristics of our predictions about that world.

3.2.3.2 Why were these parameters chosen?

The perceptual and cognitive strategies above place a high value on novel stimuli. Causes which precede their effects by more than a couple of clock ticks are not attended to. In the world described above, this is perfectly reasonable behavior. If the world had behaviors in it where more prior history was important, it would be necessary to attend further back in time to make schemas which accurately predicted the effects of actions.

The actual temporal horizons[13] used were determined empirically. For example, several runs using horizons for parameters of 1, 2, 3, 5, 10, 20, and 100 were tried, for both horizons (in other words, the cross-product of most of the combinations), and it was observed that the increase in learning was negligible (though not zero) above the values chosen. In the particular case of the sensory update horizon, values smaller than two tended to cause the statistical machinery to malfunction, missing most transitions (e.g., it became difficult to perceive that some particular bit did *not* change after some particular action).

Those readers familiar with the full schema mechanism described in [Drescher 91] may wonder about the interactions of synthetic items and this temporal horizon, particularly synthetic items employing composite actions. The runs investigated did not tend to generate large numbers of synthetic items (although see Section 4.4.2, on page 87, for some other remarks on this subject). If synthetic items *with composite actions* (which are used, for example, to represent object persistence [Drescher 91] and whose values might change

13. As specified in Section 3.2.3, the horizons were two clock ticks when doing perceptual update, and one clock tick when finding candidates for spinoffs.

due to events arbitrarily far in the past) were much more common, the temporal horizons used would most likely require some adjustment upwards. However, as Drescher points out [Drescher 94], it might suffice to represent intermediate states that keep track of the effects of the past events, so that only the temporally local values of those intermediate states need be attended to. However, this has not been investigated here.

Note that the above problem with synthetic items and the temporal horizons would *only* be true, however, if such synthetic items had actions which were composite—and this implementation, which lacks composite actions, cannot ever generate such synthetic items. Since all synthetic items generated in *this* implementation are therefore noncomposite, looking arbitrarily far back in history is not necessary.

These particular strategies also place a high value on a very specific spatial locality. Even sensory items that are very near items which have changed are not attended to. Since this microworld only has objects which are one bit wide, and the actions which involve them are, e.g., touch (which requires contact), this strategy works well.[14] In a world where actions had effects farther away than a single pixel in the visual field, or which contained objects subtending more pixels in the visual field, for example, such a strategy would have to be modified.

14. Selectors which attended to the unchanged items in a spatial "halo" around changed items were found to be less efficient, in terms of work per reliable schema, than the selectors described here. A different microworld (such as one with spatially larger objects, or different types of actions available to the agent) might require selectors that attended to a wider radius of (unchanged) sensory items around items which actually changed, in which case such "haloing" would be necessary to reliably learn the effects of actions.

3.3 Results

3.3.1 Evaluation of the focused algorithm

Two crucial questions that must be addressed concern how much the system learns *(completeness)*, and whether what it learns actually reflects its experience *(correctness)*. We must evaluate whether these focus of attention mechanisms impair that learning in any way. After all, one way to decrease the work of learning would be to simply ignore the world completely—but the resultant gain in speed could hardly be said to be worthwhile, because nothing would be learned.

The discussion below is based on results from the infant/eyehand scenario. Results from the Hamsterdam scenario have been comparable. However, since the majority of the goal-independent mechanism was developed using the infant/eyehand scenario, the Hamsterdam results were primarily restricted to simple verification of similar savings for similar mechanisms, and a relatively smaller number of experiments were performed. (The infant/eyehand scenario had literally dozens of experimental runs performed while the algorithms were being developed, from which, e.g., the chart shown in Figure 8, on page 55, is only a small sample. The Hamsterdam scenario tried out the resultant algorithms for verification and demonstrated that they worked acceptably there, too, but was not as exhaustively sampled as was the infant/eyehand scenario.)

3.3.1.1 Completeness

The schema system generates thousands of schemas in runs of reasonable duration, for instance, runs of ten or twenty thousand iterations have generated over 7000 schemas. How is one to know what all of these facts really represent? The state of the knowledge base is critically dependent upon prior knowledge: a more-detailed schema can only be generated from a less-detailed one, so any change in the learning mechanism which changes which

schemas are generated leads to rapidly-diverging sets of generated schemas. While all may say *approximately* the same thing, the fine details of exactly which facts are learned will tend to be different. It would be possible to run enough iterations so that almost every possible fact that *is* true about the microworld could be *learned* to be true, but this is an unreasonably large amount of computation (the total number of learned schemas plotted over time appears to have an asymptote at least in the tens of thousands of schemas, even for this simple world).

3.3.1.1.1 Manual evaluation methods

Manual inspection of the schemas generated by these runs was employed as a first cut at establishing that alternative focus mechanisms were not substantially decreasing completeness, and tools were developed for examining how many schemas, representing what general categories of facts (e.g., unimodal visual field, multimodal across various modalities, etc) were being learned. By comparing rough totals of different types of generated schemas, one could obtain at least some assurance that some particular class of schemas was not being systematically omitted.

Another manual method of checking the results employed *n*-way comparisons of the generated schemas themselves. The **(context, action, result)** triple of each schema can be represented relatively compactly in text (ignoring all the statistical machinery that also makes up a schema); by sorting the schemas generated in any particular run into a canonical order, and then comparing several runs side-by-side, one can gain an approximate idea of how different runs fared. Figure 7, on page 52, demonstrates a tiny chunk from a 5-way comparison of a certain set of runs, in which 5 somewhat-different runs were compared for any large, overall changes to the types of schemas generated.[15]

3.3.1.1.2 Automatic evaluation methods

Manual methods are tedious and error-prone. Furthermore, the underlying reason that an agent learns is to aid it in the pursuit of its goal. This means that a sensible evaluation strategy is to ask if the agent has, indeed, learned enough to accomplish goals that it was unable to accomplish before learning.

A simple way to establish what the agent knows is therefore to use the generated schemas as parts of a plan, *chaining* them together such that the result of one schema serves as the context of the next, and to build these chains of schemas until at least one chain reaches from the initial state of the microworld to the goal state. If we can build at least one such chain, we can claim that the agent *knows* how to accomplish the goal in that context; the shortness of the chain can be used as a metric as to *how well* the agent knows.[16] For this task, the schemas to be used should be those deemed *reliable*, e.g., those which have been true sufficiently often in the past that their predictions have a good chance of being correct. Simply employing *all* schemas, reliable or not, will lead to many grossly incorrect chains. (A more complete description of the chaining system, and its use in evaluating the results of learning, is deferred until Chapter 4, where the generated chains are also used in goal-directed learning and behavior.)

At the start, no facts about the world are known, hence no chain of any length can be built. However, after a few thousand schemas are built (generally between 1000 and 5000),

15. The layout of this chart, and its ordering, is not accidental. This is, after all, a manual evaluation method; it depends on the ability of the human visual system to pick out aberrations in patterns. Large holes or gaps in the columns merit closer attention, allowing the effort of checking carefully to be limited to only a small number of cases.

16. Note that the small size of the microworld and the small number of actions possible at any given timestep mean that even a random walk through state space has a significant chance of accomplishing the goal, if we are willing to wait long enough; hence, a path which is *close to optimal*, rather than one which exists at all, should be our metric for whether learning has succeeded. See Section 4.4.3, on page 91, for a comparison of the length of the chains built for a typical goal versus the average length of a random-walk to the goal.

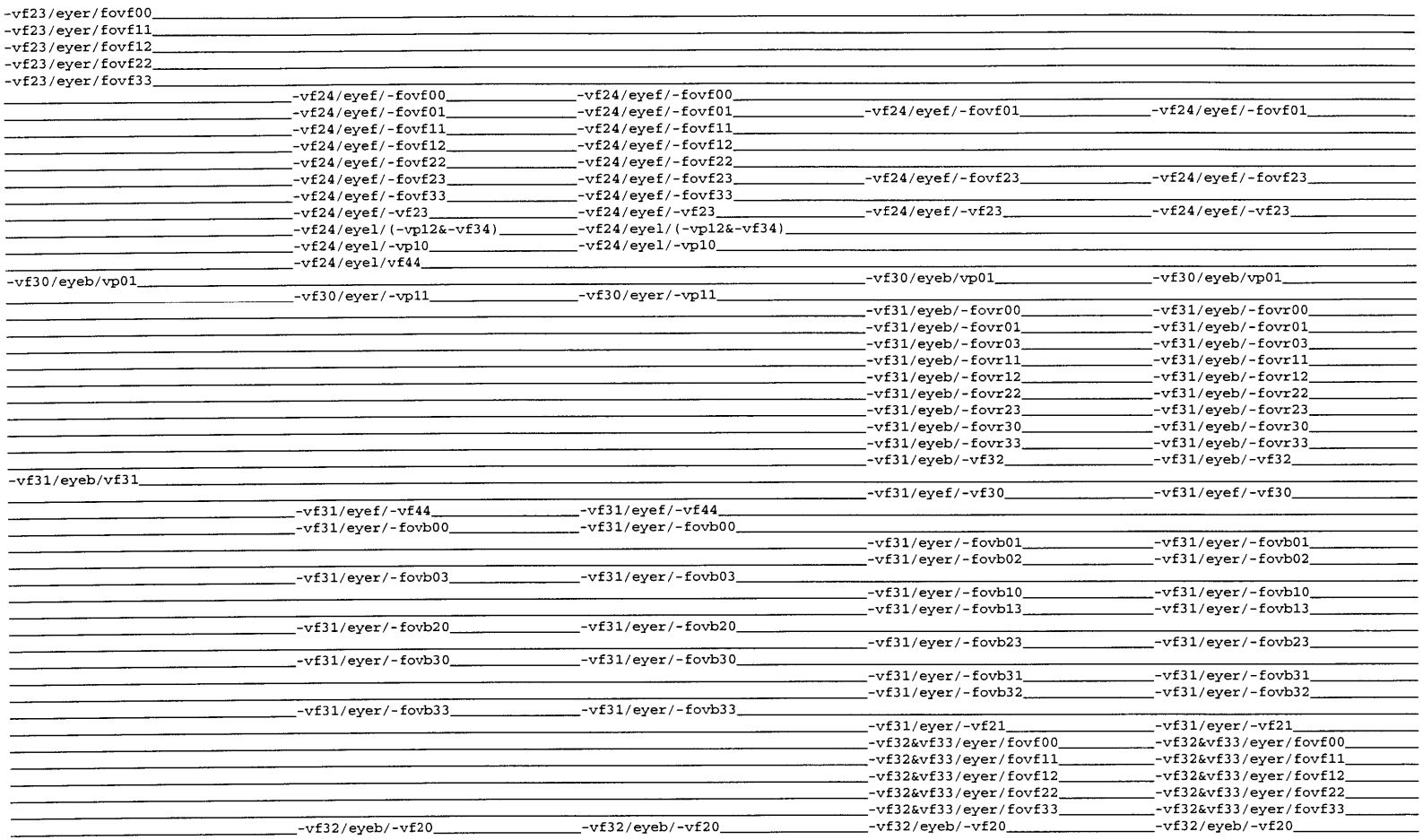

Figure 7: A tiny chunk of a 5-way comparison of generated schemas

Five columns of schemas, sorted alphabetically by their printed representations, are shown here side-by-side, horizontally aligned. Small holes are fine, but large holes could indicate a potentially missing class of schemas.

most starting states can plausibly chain to a simple goal state, such as centering the visual field over the hand, in a close-to-optimal number of steps.

3.3.1.1.3 Completeness results

Given this mechanism, how well did the focus of attention mechanisms fare? Quite well. In general, given the same approximate number of generated schemas, both the basic and focused approaches cited above learned "the same" information: they could both have plausibly short chains generated that led from initial states to goals. Both the chaining mechanism described above, and manual inspection, showed no egregious gaps in the knowledge or particular classes or types of facts that failed to be learned.

As shown in Figure 8, on page 55, and explained in Section 3.3.2, on page 54, the focused approach tended to require approximately twice as many timesteps to yield the same number of schemas as the unfocused approach. This means that a real robot which employed these methods would require twice as many experiments or twice as much time trundling about in the world to learn the same facts. However, the reduction of the amount of computation required to learn these facts by between one and two orders of magnitude[17] means that the processor such a robot must employ could be much smaller and cheaper—which would probably make the difference between having it onboard and not. This is even more compelling when one realizes that these computational savings get bigger and bigger as the robot learns more facts.

3.3.1.2 Correctness

The statistical machinery of the schema mechanism goes to great pains to avoid being fooled by occasional coincidence. Only if some change in the state of the world is positively correlated with an action more often than it is negatively correlated, and only if we have seen enough instances of both the event happening after some specific action and the

17. For runs of this length, e.g., 1000-2000 schemas generated.

event *not* happening in the absence of the action, *and* if the event is unexplainable by any other schemas, will the mechanism conclude that the action is truly the cause of the event. (This is an overview of the marginal attribution mechanism described in Section 3.1.3.2, on page 34, and in [Drescher 91].)

Because of this, the only way that any learning algorithm which uses this system could learn "incorrect" facts (e.g., correlations that do not, in fact, reflect true correlations in the world) would be to *systematically* exclude relevant evidence that indicates that a schema that is thought to be reliable is in fact unreliable. No evidence of this was found in spot checks of any test runs. It is believed (but not proven) that none of the focused algorithms described can lead to such systematic exclusion of relevant information: the mechanism may miss correct correlations (such is the tradeoff of having a focus of attention in the first place), but it will not miss only those correlations that would tend to otherwise invalidate a schema thought to be reliable.

3.3.2 Comparison of the different strategies

Figure 8, on page 55, presents partial results from several runs with different choices of selectors. Only the most salient combinations of selectors were included in this table. Of those, the rows in boldface will be discussed below; the non-boldfaced rows are included to give a feel for how different choices can influence the results.

The results in this table were all produced by runs 5000 iterations long. Similar runs of two or three times as long have produced comparable results, with correspondingly greater increases in β (see below).

The first four columns of the table show the particular selectors in use for any given run; the top row shows those selectors which correspond to the basic (Drescher) algorithm,

Algorithm				Learning			Work required			Facts per work unit		
Spinoff selectors		Statistic selectors		Schemas			Inner loops (x10^6)			Reliable schemas over		
Items	Schemas	Items	Schema	Total	Rel	T/R	Spin	Stat	Both	Spin	Stats	Both
AIN	**ASN**	**AIN**	**ASN**	**1756**	**993**	**1.77**	**533**	**533**	**1066**	**1.9**	**1.9**	**0.9**
AIN	ASN	CINIH	ABSPSDUCI	1135	403	2.82	398	12	410	1.0	33.6	**1.0**
AIN	ASN	AIN	ABSPSDUCI	1110	518	2.14	391	55	446	1.3	9.4	**1.2**
CIN	**ASN**	**AIN**	**ASN**	**1693**	**948**	**1.79**	**44**	**524**	**568**	**21.5**	**1.8**	**1.7**
CIN	**SWRUS**	**AIN**	**ASN**	**1395**	**791**	**1.76**	**2**	**463**	**466**	**316.4**	**1.7**	**1.7**
CIN	ABSPSDUCI	AIN	ASN	1622	924	1.76	15	510	525	61.6	1.8	**1.8**
CIN	ASN	AIN	ABSPSDUCI	1110	506	2.19	33	54	87	15.3	9.4	**5.8**
CIN	ABSPSDUCI	AIN	ABSPSDUCI	1110	506	2.19	10	54	64	50.6	9.4	**7.9**
CIN	ABSPSDUCI	CINIH	ASN	1366	643	2.12	13	64	77	49.5	10.0	**8.4**
CIN	ASN	CINIH	ABSPSDUCI	1136	399	2.85	34	12	46	11.7	33.3	**8.7**
CIN	**SWRUS**	**AIN**	**ABSPSDUCI**	**1102**	**498**	**2.21**	**1**	**53**	**54**	**415.0**	**9.4**	**9.2**
CIN	SWRUS	CINIH	ASN	1353	688	1.97	2	64	66	275.2	10.8	**10.3**
CIN	ABSPSDUCI	CINIH	ABSPSDUCI	1136	399	2.85	10	12	22	40.7	33.3	**18.3**
CIN	**SWRUS**	**CINIH**	**ABSPSDUCI**	**1134**	**398**	**2.85**	**1**	**12**	**13**	**331.7**	**33.2**	**30.2**

Figure 8: Summary of goal-independent results

The names for the algorithms used in learning are explained in Section 3.2.1 through Section 3.2.3. Results from the rows in boldface are discussed in this section. The non-bold-faced rows are not discussed in the text, but are included for additional context. This table is a sampling; in all, in excess of 30 different combinations of selectors were investigated.

The top line is effectively the "unfocused" case, as in [Drescher 91]; the bottom line is considered the "best" or most tightly-focused case.

while the bottom row shows the most highly-focused algorithm, as described in Section 3.2.3, on page 44.

The table is sorted in order by its last column, which shows number of *reliable* schemas generated during the entire 5000-iteration run, divided by the amount of total work required. For conciseness, we shall call the value in this column β, which is defined as:

$$\text{Facts per work unit} = \beta = \frac{SchemaRel_{Total}}{Work_{Total} \times 10^{-6}}$$

where the multiplication by 10^{-6} is simply to normalize the resulting numbers to be near unity, given the millionfold ratio between work units and number of schemas generated.

The bold rows in the table show successive changes to the selectors used, one at a time. The top row is the basic algorithm, which shows that about a billion total inner loops were required to learn 1756 schemas, 993 of which were reliable, which gives a β of 0.9.

Note that, because the world is stochastic (for example, the two "inanimate" objects occasionally move from one square to a neighboring square, approximately every few hundred clock ticks), one might imagine that there would be variance in the number and reliability of schemas generated between two runs, even if they use the same strategy. In fact, this is true, but the variance is quite low: out of a run of two or three thousand schemas with the same strategy and different seeds for the random number generator (hence different random behaviors in the world), the difference in the number of schemas generated is generally less than ten. In other words, the number of schemas generated is usually within 1% between runs using the same algorithm. Further, the types of schemas generated also match each other quite closely, as determined by *n*-way comparisons between runs, using the techniques discussed in Section 3.3.1.1.1, on page 50. (The exact schemas generated will, of course, be different, as discussed in Section 3.3.1.1, on page 49).

Let us examine changes to the selectors for spinoffs, which determine the cognitive selectivity, or what is attended to in learning new schemas from the existing schemas. When we change $Spin_i$ from **AIN** to **CIN** (e.g., from all item numbers to those whose items changed at the last clock tick), the amount of work drops by about a factor of two, while the number of generated schemas barely decreases. This means that virtually all schemas made predictions about items which changed in the immediately preceding clock tick (e.g., that corresponding to the action just taken), hence looking any further back in time for them costs us computation without a concomitant increase in utility.

Changing $Spin_s$ from **ASN** to **SWRUS** (e.g., from all schema numbers to those whose statistics were recently updated), given that $Spin_i$ is already using the selector **CIN**, yields a small improvement in β (not visible at the precision in the table), and also a small improvement in the ratio of reliable to total schemas. (Were $Spin_i$ not already **CIN**, the improvement would be far more dramatic, as demonstrated in runs not shown in the table.) Note, however, the enormous decrease in the amount of work done by the spinoff mechanism when $Spin_s$ changes from **ASN** to **SWRUS**, dropping from 10% of $Work_{Total}$ to 1%.

Next, let us examine the effects of perceptual selectivity. Changing $Stat_s$ from **ASN** to **ABSPSDUCI** (e.g., from all schema numbers to all bare schemas plus schemas dependent upon changed items) increases β by a factor of 5.4, to 9.2, by decreasing the amount of work required to update the perceptual statistics by almost an order of magnitude. In essence, we are now only bothering to update the statistical information in the extended context or extended result of a schema, for some particular sensory item in some particular schema, if the schema depends upon that sensory item.

Finally, examine the last bold row, in which $Stat_i$ was changed from **AIN** to **CINIH** (e.g., from all item numbers to all item numbers whose items changed in the last two clock ticks). β increases by a factor of 3.2, relative to the previous case, as the amount of statis-

tical-update work dropped by about a factor of four. We are now perceiving effectively only those changes in sensory items which might have some bearing in spinning off a schema which already references them.

Note that each successive tightening of the focus has some cost in the number of schemas learned in a given number of iterations. This means that, e.g., a real robot would require increasingly large numbers of experiments in the real world to learn the same facts. However, this is not a serious problem, since, given the focus algorithm described here, such a robot would require only about twice as many experiments, for any size run, as it would in the unfocused algorithm. This means that its learning rate has been slowed down by a small, roughly constant factor, while the computation required to do the learning has dropped enormously.

Chapter 4: Goal-dependent Learning

4.1 Introduction

This chapter discussed the influence of goals on reducing the work of learning. First, in Section 4.2 below, it summarizes the fundamental impact of goals in the learning system used here, and motivates the relation of goals to planning. In short, and as summarized in Section 4.2 and explained at length in Section 4.3.1.1, on page 64, the key ideas in using goals to focus are:

- Every goal has some built-in percepts and actions associated with it.
- Learning is restricted to those percepts and actions.

In Section 4.3, on page 63, we cover in depth the general concepts that will be required to evaluate the results presented later. In Section 4.3.1, we go into much greater detail about goals as they are used here, including their structure and more precisely how they are used to focus the learning process. Then, in Section 4.3.2, on page 71, we discuss how to evaluate the learning that has occurred, with emphasis on using planning to search the space of generated schemas and so prove, via successful construction of plans, that the schema system has indeed learned a relevant set of schemas for the given goals.

Finally, in Section 4.4, on page 86, we present some results from this paradigm, using the terminology and mechanisms described in the aforementioned sections to describe the amount of work required to learn, and the nature of the learning that has taken place.

4.2 Why goals?

Goal-independent filtering is often insufficient to accomplish effective learning. Without goals, it is difficult to know what would be useful to learn, hence there is no way to know a priori whether some particular fact needs to be learned or not. This leaves an agent with the task of trying to learn everything, just in case it will be needed later.

This chapter investigates one way of adding goals to the learning process. The aim is to reduce the amount of work performed, by reducing the scope of learning from every correlation that appears in the microworld to those correlations that seem relevant for some particular set of goals.

Using goals allows us to define two sets which constrain the amount of work performed, in a similar manner to the way in which the work of learning was reduced in Chapter 3; furthermore, both the methods of Chapter 3 and the methods described in this chapter may be combined, to decrease the work of learning in two ways at once.

For any given goal, the set of *percepts* defines those sensory items which this goal allows the learning system to attend to. When updating item statistics, only these items are updated; when deciding which schemas to spin off, only schemas dependent upon (e.g., making predictions about) those items are noticed. This same mechanism is used in a goal-independent fashion in Section 3.2, on page 40.

Similarly, the set of *actions* defines which actions are allowable when this goal is active. Typically, this set is "all actions" or "all eye motions" or "all hand motions," etc.

This constraint of percepts and actions is the central mechanism in this research by which goals decrease the effort of learning; the rest of this chapter details how the mechanism works, and how to evaluate it.

Note that the word *goal* can mean many different things in different contexts. A goal could be the final, end result desired from a long series of actions (e.g., strategic), or the next small result that one desired (e.g., tactical), or something in between. This research treats the word "goal" more as a tactical, e.g., short-range, pursuit, and *chains* such tactical goals together to make larger, strategic or long-range, pursuits. This is explained in more detail in Section 4.3.1.2, on page 68, when we show some representative goals used in this system.

4.2.1 Where do goals come from?

In real creatures, goals may come from a variety of sources. Simple creatures (such as insects, small mammals, etc) are generally directed essentially completely by instinct. In this case, the animal's strategic goals are generally motivated by homeostasis concerns (e.g., not getting get too hungry, thirsty, hot, or cold), and genetic concerns (e.g., mate and/or care for offspring in the right season, etc). Its tactical goals are motivated by a combination of these strategic goals and its immediate environment and situation (e.g., if the animal is hungry and there is food nearby that appears edible, it eats it).

More complicated creatures have more complex sets of goals, enabled in general by more complex and capable sensory systems and cognitive mechanisms of greater sophistication. As we walk up the phylogenetic tree towards human-level performance, the influence of instincts on choosing and following goals diminishes. Such more complicated animals have more cognitive autonomy in choosing their own goals, formulating new goals, and deciding how to carry them out. At the human cognitive level, we can confidently assert that we can reason about our goals, use introspection to evaluate them, and so forth. The mapping between some tactical or strategic goal and the resulting behavior has become more and more divorced from a simple stimulus/response paradigm, and the

immediate environment plays less and less of a role in determining what the organism's next action or next cognitive procedure shall be.

This research does not attempt to model such a high level of competence. Here, we treat goals as they might be in an insect or simple animal. It is presumed, therefore, that the mapping from a goal to what actions might be performed, or what aspects of the world shall be attended to, is *hardwired* into the organism; in a real organism, such hardwiring is presumably accomplished evolutionarily, by favored survival and reproduction of creatures for which the correct mapping of goal and current situation to cognitive focus and action existed. Similarly, the complete set of possible goals is fixed in advance; the agent cannot *invent* a new goal for itself.

In addition, the toplevel goal for any given run (whether it is a *learning* run, in which case new schemas are being generated, or a *performance* run, in which schemas that were generated earlier are being evaluated to determine the agent's competence) is chosen at the beginning of the run, and is thus outside of the agent's control.

Goals and the ways in which we use them, both for learning and for evaluation of what learning has taken place, are described in much more detail in Section 4.3.1, on page 63.

4.2.2 The relation of goals to planning

Pursuit of a strategic goal generally implies attempting a sequence of behaviors that is intended to lead to the goal's successful completion. Since all but trivially simple worlds generally require executing a series of such behaviors to get from the initial state to the goal state, goal-oriented behavior leads immediately into the realm of *planning*.

The literature on planning is immense; this research does not explicitly attempt to extend it. Instead, it uses some results from that prior art to implement goal-directed focus of attention: in particular, it plans chains of tactical goals from some starting state to some

strategic goal state. Plans and the ways in which we use them, both for learning and for evaluation of what learning has taken place, are described in more detail in Section 4.3.2, on page 71.

4.3 General concepts

The two most important concepts that must be understood in how we use goal-directed learning are those of goals and plans. Goals and plans are used both to guide the learning process, and to evaluate, after some learning has taken place, how *competent* the agent is, e.g., whether or not it has learned any useful information about the results of its actions in its environment.

4.3.1 Goals

As used here, a *goal* is primarily composed of a set of *percepts* (which determine which sensory inputs will be attended to, and which schemas will be available for lookup in the memory) and a list of permitted *actions* (which determine which actions are allowed). Using this mechanism, it is possible to focus sensory and cognitive attention to a particular subset of all possible inputs for a given goal, and it is also possible to restrict the actions available for execution to those that might be useful for the goal.

There is little distinction in this system between strategic and tactical goals. In general, goals are treated as tactical (e.g., each one defines a relatively simple desired change from the current perceived state of the world to the intended state), and these tactical goals are chained together to make more complicated and long-range goals. However, the system does not treat a chain of such goals as some more complicated (strategic) goal in and of itself, and has no representation for "a goal composed of several subgoals." Instead, it rep-

resents the set of tactical goals in any given run as a finite-state machine, executing the transition from one tactical goal to another when each such goal claims that it is satisfied.

When talking about evaluating the competence of a set of schemas (in order words, did the schemas learned enable the system to reach some particular goal),to *reach the goal* denotes some particular tactical goal which is distinguished as *the goal.* This goal is like any other goal in the system, except that it also tells the FSM that it is time to record a success and to try another test goal. Hence, to "reach the goal" is terminological shorthand for to "reach the strategic goal," which is itself short for "to reach the goal which has the distinguished marker stating that this is the end goal, whose satisfaction should be recorded as a success in accomplishing some strategic pursuit."

The FSM evaluates the various parts of the active goals,[1] and decides which new goal(s) should become active, at each clock tick; thus, for each clock tick, we take an action from the set of actions permitted by the currently-active goals, update the internal representation of those sensory bits that the active goals allow us to observe, run the learning system one clock tick, and advance the FSM one transition.

4.3.1.1 The structure of goals

There are several pieces or *slots* which make up a complete goal. The complete structure of a goal, excluding certain implementation-specific slots used for internal housekeeping, is therefore as follows. Remember that these goals are essentially *tactical* goals, and that each goal contains the information inside it which specifies which tactical goals may become active when it declares itself to have succeeded.

- *Name.* The name of this goal. (This is how one goal refers to some other goal, and how the programmer can specify which goal is which.)

1. There can be more than one goal active at a time; see the next section.

- *Percepts*. Set of which items that this goal allows the learning system to attend to.
- *Actions*. Set of which actions this goal allows the learning system to take.
- *Concurrent*. Which other goals to pursue at the same time. (These other, concurrent goals add their percept and action sets in to those defined in this goal, and the union of all of them makes up the set of percepts and actions that are attended to at the moment.)
- *Next*. Which other goal (singular!) to pursue next, if this goal claims to have succeeded on this clock tick.
- *Lose*. Goal to (maybe) transition to if this goal does not succeed on this clock tick.
- *Win*. A function called to determine this goal's success (used while learning).
- *Schemas-final*. A function called to determine membership in FINAL (used while chaining to goals; the meaning of FINAL will be defined in Section 4.3.2.5.4, on page 81).

The set of *percepts* defines those sensory items which this goal allows the learning system to attend to. When updating item statistics, only these items are updated; when deciding which schemas to spin off, only schemas dependent upon (e.g.,making predictions about) those items are noticed. This same mechanism is used in a goal-independent fashion in Section 3.2, on page 40.

Similarly, the set of *actions* defines which actions are allowable when this goal is active. Typically, this set is "all actions" or "all eye motions" or "all hand motions," etc.

The WIN slot for a goal, if specified, contains a function which is run to determine whether or the goal is considered to have succeeded. This function is particular to each

goal, and generally specifies some combination of sensory items whose state must be on, off, or don't-care for the goal to succeed.

Goals can *inherit* from other goals; this is a way of easily and modularly building up more complex goals from simpler goals. If goal A has a *concurrent* slot that mentions goal B, then A inherits the percepts and actions of B, increasing the number of possible percepts and actions that A has available. (In other words, inheriting from another goal can only increase the number of sensory bits attended to, or the number of permitted actions, and can never decrease it.)

Suppose, as above, that goal A inherits from goal B. The inheritance is actually managed by running both goals *in parallel*; there is nothing preventing A from inheriting from B while B is inheriting from A. Both of their WIN slots, for example, will be evaluated, and whichever one wins chooses the NEXT slot that determines which next goal state will result.

If, in this case, A claims to have won, and B does not, then its choice of the next goal completely overrides whatever B's NEXT slot might claim. The FSM transitions to the goal named by A's NEXT slot, completely abandoning A and any other concurrent goals that were running at the same time (e.g., B).

If both A and B both claim to win during the same clock tick, then the real winner (the one whose NEXT slot chooses the next goal) is undefined; one or the other will be chosen arbitrarily.

The reason that this parallelism is called "inheritance" stems from what happens if B, to use the example above, fails to mention *any* WIN, NEXT, etc, slots. In that case, B cannot possibly influence *which* goal is picked next, but only what sensory items are attended to or actions are possible; A has *inherited* B's "focus."

Given the above, we run a finite-state-machine, using the goals as a simple rule-based system—the goals determine under what conditions the FSM will transition to the next state, and what state that will be. First, we call the WIN slot, to see if the goal is satisfied or not. Once we have decided whether the goal is satisfied or not, we have two choices. If the goal was satisfied, then we transition to the goal named in the NEXT slot for this goal. If the goal was *not* satisfied, then we remember the goal named in the LOSE slot, and we try any other concurrent goals that may exist. (Note that the order in which we try concurrent goals is undefined and should be viewed as "simultaneous"—this is why it is undefined which goal's NEXT slot will be used if they both claim to win.) If none of them succeed, then we pick the next goal randomly from the goals in the LOSE slots. If none of the concurrent goals had any LOSE goals defined, then we stick in the current goal. (This allows us to define a goal that we cannot come out of, assuming no concurrent goals are being run with it, by failing to define anything for the LOSE slot. On the other hand, we can avoid getting stuck by making sure that at least one concurrent goal has this defined.)

The actual implementation of the FSM piggybacks somewhat upon the implementation of goal-independent learning described in Section 3.2, on page 40. In particular, the iterators that determine which schemas and sensory items are attended to (see Section 3.2.3, on page 44) are *masked* by the set of active percepts and allowed actions, such that the existing goal-independent iterators may be reused. The masking action allows the goal-dependent part of the algorithm to use the logic from the goal-independent part of the system untouched; adding goals will never increase the number of attended items or schemas, and usually decreases it. However, using goals does not necessary mean that the goal-independent system must be running in the most tightly-focused mode as described in Section 3.2.3, on page 44; in particular, were one to wish to, one could run the goal-dependent system while using the full-crossbar (e.g., most inefficient) mode of the goal-indepen-

dent system. (There seems little or no reason to ever want to do so, but it would certainly be trivially possible to do so.)

There is an important additional detail about the way the FSM runs. It is not *always* the case that the action taken is drawn from the set of goal-dependent currently-permitted actions. A fixed percentage of the time (at the moment, 10%), the set of allowed actions is expanded to include *all* possible actions. Why is this done? Because, especially in the case of the infant/eyehand microworld, the microworld is relatively static. (So-called inanimate objects move around every few hundred clock ticks, and the hand never moves unless commanded.)

Consider what would happen in, e.g., the infant/eyehand scenario if the goal system did not occasionally allow a completely random action to take place. Suppose that the goal set being run was one which tried to learn how moving the eye affects what is seen, and that the hand was never allowed to move. This relatively-static microworld would otherwise falsely teach the learning system that, e.g., the hand is *always* in some particular position, because the agent was never allowed to move it. When it then came to take action based on what the agent had learned, this belief that the hand never moved would cause many plans to go awry.

Thus, in worlds in which nothing much happens unless it is in response to an action taken by the agent, it is important to occasionally allow the agent to take an action which seems unrelated to what it is currently trying to learn—otherwise, it may falsely learn things it believes to be *always* true which are instead simply true *when the agent is not permitted to take some particular action.*

4.3.1.2 Using goals for learning

A typical use of these tactical goals for learning is to concentrate effort on a particular strategic goal deemed *useful* for some reason.[2] In this case, such a strategic goal would be

"center an object in the visual field," "identify what object is in the visual field" (which requires at least getting it into some foveal region, where enough detail is available to do so), "grasp an object," and so on.

A typical set of goals which comprise an FSM for identifying an object appears in Figure 9, on page 69. This FSM was designed for the infant/eyehand scenario; the Ham-

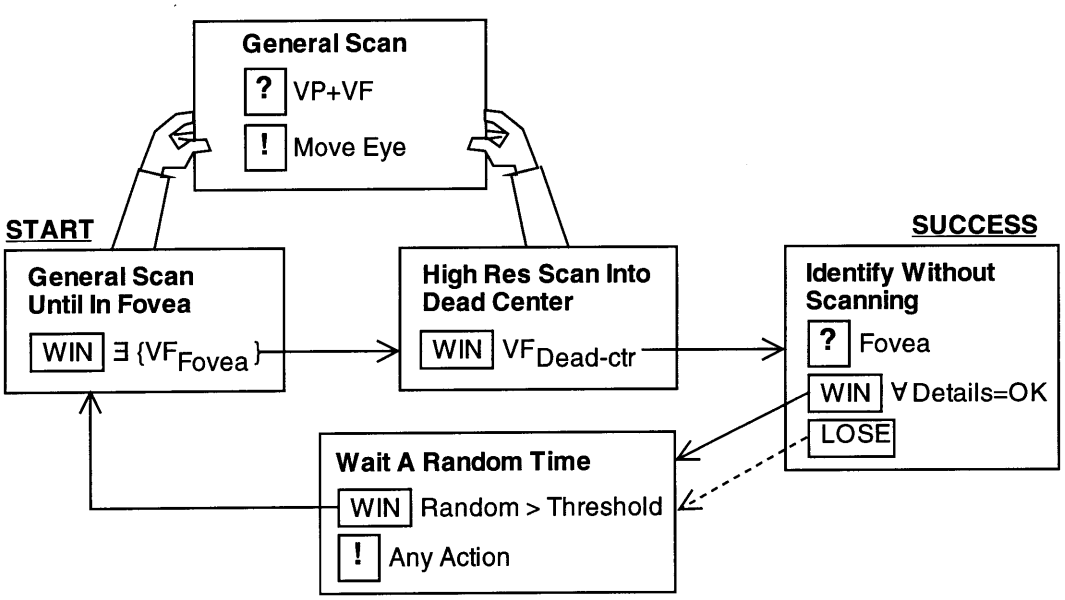

Figure 9: A typical set of goals

sterdam FSM for the same task is similar. In this chart, percepts are denoted by a "**?**" and permitted actions by "**!**". The use of VP+VF in the percepts for *General Scan*, for example, indicates that visual-proprioceptive and coarse-visual sensory items should be attended to, and the rest ignored (assuming, of course, that no other concurrent goal adds to this set).

VF_{Fovea} and $VF_{Dead\text{-}ctr}$ denote a WIN function that succeeds if any sensory item corresponding to "somewhere in the fovea" or "dead-center in the fovea" is on, respectively. Similarly, the WIN slot in the goal *Identify Without Scanning* specifies that all the details

2. Such strategic goals are currently declared useful by the programmer, not by the system.

(from the fine-foveal bits) of the object must match some target set of details for the goal to declare that it has succeeded.

The FSM shows that the goals *General Scan Until in Fovea* and *High Res Scan Into Dead Center* both inherit percepts and actions from *General Scan* (the hands show the inheritance pathway). When *General Scan Until in Fovea* succeeds, the FSM transitions to *High Res Scan Into Dead Center*; when that succeeds, the FSM transitions into *Identify Without Scanning*, which finally transitions to a state which simply waits a random amount of time to let the world random-walk away from the goal state.[3] Then we repeat the exercise.

Expressed in a more general fashion, the idea in this goal set is to notice an object somewhere in the (coarse) visual field, scan until it is in the fovea (in this goal, we first scan until it is somewhere in any coarse visual item corresponding to the position of the fovea, then scan with the foveal fine-detail bits enabled while we try to get the item dead-centered in the fovea), then to attempt to identify it using the foveal detail sensory bits.

This FSM is actually slightly more than is necessary; for example, in the two microworlds used here, having an object anywhere in the foveal region is sufficient to get enough detail to identify it. However, in order to make the goal somewhat harder and more interesting to achieve (given that the area occupied by the fovea is relatively high in both microworlds), we insist that the object be exactly centered before attempting to identify it. (In the Hamsterdam scenario, instead of insisting that the object be "centered," we insist that it be "in the center foveal ray and at very short range.")

3. See Section 4.3.2.7, on page 83, for why this random walk is necessary. In short, the idea is to allow the state of the world to decay away from the strategic goal, by allowing random events to transpire. If we did not do this, then we would immediately try to get to that goal again, and presumably succeed immediately. This would not be very informative.

4.3.2 Evaluating learning

The simplest way to evaluate what the schema system has learned is to use the same methods as in Chapter 3, namely, to count schemas and to count the work required to generate them. In Section 4.4.1, on page 86, we do exactly this for a typical example of a goal-directed run. But what exactly does this mean? After all, with the introduction of goals, entire swaths of domain knowledge may be purposely omitted, in order to concentrate computational effort on different areas of the domain—consider, for example, that there will be no schemas in the example above which talk about hand motion, since none of the goals in the run allow movement of the hand, nor do any of those goals allow the perception of any hand-related sensory items.[4] Consider also that the number of conjunctions and synthetic items generated in a goal-directed run is much larger than that for a non-goal-directed run (see Section 4.4.2); what are we to make of this?

In Section 3.3.1, on page 49, we glossed over the issue of exactly what the schema system had learned in various configurations. The introduction of goals makes such evaluation more problematic. Different choices of goals lead to different information being learned. The techniques used in Section 3.3.1, can no longer be assumed sufficient a priori; instead, in this section and succeeding sections, we shall examine the issue of what the system has learned much more rigorously.

Note that the planning process described in this section is sufficient to understand the results presented below in Section 4.4. However, a naive implementation of the planner described quickly runs afoul of many $O(n^3)$ or even exponential-time complexity blowups. Appendix B briefly describes some strategies for dealing with these problems in order to

4. Even though Section 4.3.1.1, on page 64 specifies that the agent is occasionally allowed to execute a completely random action, unspecified by any goal, the set of allowable percepts is not similarly expanded. Thus, though the hand in this example may randomly be allowed to move, hand-related items such as haptic-proprioceptive or tactile senses will not be perceived anyway.

do planning in a reasonable amount of time; it may safely be skipped without loss of continuity.

4.3.2.1 Using chaining to evaluate learning

Instead of the more qualitative methods described above, here we turn to simulation to evaluate the knowledge expressed by a collection of schemas. The simulation consists of putting the schema system in a particular state of the microworld, specifying a goal, and evaluating how often there is sufficient stored knowledge to predict how to reach the goal from the starting state.

There are several ways in which one might proceed. We examine some of the options here. Next, we look at whether or not to learn while performing the evaluation, and later sections examine some details of running the simulations themselves.

4.3.2.2 Should we learn during evaluation?

Should we learn additional schemas while attempting to reach the goal? In the work reported here, this is *not* the case; the learning part of the schema system (e.g., updating statistics of sensory items, and spinning off new schemas) is inhibited during the simulation or evaluation phases. This is in part to keep from muddying the waters (continuing to learn would mean that, the longer a simulation ran, the more knowledge would be available for it or for future runs with different parameters, complicating comparison), and in part for issues having to do with speed of evaluation (certain techniques used in the evaluation would be more difficult or run much more slowly if learning during the simulation was allowed; in particular, many of the caching strategies described below in Appendix B would fail).

In general, of course, a real agent should not suppress learning while attempting to reach a goal; this is done here simply as a convenience. There are also other solutions than the sort of planning involved here; for example, the composite action mechanism of

[Drescher 91], especially when combined with the intrumental and delegated values mechanisms defined there, shows one way to plan paths to goals that is particularly efficient on parallel hardware.

4.3.2.3 When to plan?

If one is planning a path from an initial state to a goal state, and if the path is longer than a single step, a natural question to be addressed is whether to replan the path at each step towards the goal.

Nonincremental, or *ballistic*,[5] planning, in which the path to be followed is planned once, at the start, and then executed open-loop until it is played out, has the advantage of being faster to compute than *incremental*, or closed-loop, planning, since the plan must only be generated once per attempt. This, of course, assumes a situation in which generating the entire plan is feasible. If the space to be planned in has an exponential computational complexity—as is almost always the case—then such open-loop planning takes a very long time. If we are allowed to plan only partway to the goal, and then replan, we start to approximate incremental planning.

On the other hand, if we plan the complete path from the current state to the goal state at *every* timestep, we can cope better with failures of the plan en route. Intermediate solutions (e.g., build a partial or complete plan, execute it for some number of steps, and check along the way for certain "disaster" indications and replan if so) are also common in the planning literature.

The approach taken here is twofold. Some runs were performed with completely ballistic planning, in which the plan was formulated and the entire sequence of actions was carried out, after which the state of the microworld was checked to see if we were in the goal state. (But see Section 4.3.2.5, on page 79, for a slight modification to this procedure which

5. By comparison with ballistic, e.g., open-loop, movements such as ocular saccades.

more accurately describes how planning was performed.) Other runs were performed with incremental planning, in which the entire plan was completely replanned at every timestep.

4.3.2.4 How to plan?

4.3.2.4.1 Chaining

When building a plan from some starting state to the goal state (often referred to here as *chaining*, e.g., building a chain of schemas which reflects, at each step of the plan, what action to take to cause one world state to be transformed into the next desired state), one must decide what validly constitutes a plan.

As described here, a plan consists of a series of schemas. The *first* schema in the plan has a context which is *satisfied*. In other words, the schema's context, for any items which it asserts either positively or negatively, accurately describes the states of those sensory items which it specifies. Don't-care items in its context (i.e., most of them) are allowed to take on any value in the world (on, off, or unknown). This is the standard definition of the *applicability* of a schema from [Drescher 91].

The *last* schema in the plan has a *context* which is considered satisfied when the world is in the goal state. In other words, the last schema is applicable when the goal has succeeded. Note that this schema's action should *not* be executed as part of the plan, since it is this schema's *context* that we are interested in; its result (which would probably obtain if we executed its action) would take us one step past the goal state into some other state. When talking about the length of a plan, however, we ignore this special "last" schema and talk only about the number of actions that would be taken to execute the plan; thus, the notion of plan length is identically equal to the number of actions we expect to take.

In the simplest case, a chain of schemas composing a plan would look something like A/FOO/B, B/BAR/C, C/BAZ/D, etc, where the result of one schema is the context of the next schema. This plan specifies, "If sensory item A from the microworld is asserted, then tak-

ing the actions FOO and BAR in that order will probably lead to sensory item C being asserted."[6]

4.3.2.4.2 Path metrics

The second major question about planning consists of evaluating whether a plan is any good. In the case of the microworlds investigated here, it is generally the case that a typical goal state might be reachable from the given starting state via several thousand paths—having 30,000 paths to choose from is quite common, for example. By almost any reasonable metric, most of these plans are quite poor:

- Many depend upon an initial schema whose context is null (meaning, "don't look at any sensory items; just assume that this plan will succeed, ignoring the prior state of the world—such a plan is called a *contextless start* or a *desperation plan*).
- Many others are improbably long or contain redundant steps ("move the eye left; now move it right; now move it left again," etc: there would be an infinite number of such plans if we allowed a schema to appear more than once in any given plan (we do not—see Appendix B); as it is, the number of such paths is merely very large.
- Many other plans contain one or more schemas with low reliability, in some cases vanishingly small.
- Other plans might contain a schema such as A/FOO/A,[7] which says that, "If sensory bit A is true, and we take action FOO, A is *still* true thereafter" —if

6. This becomes less simple when either contexts or results are conjunctive. Are plans such as A/FOO/B&C, B/BAR/D or A/FOO/B&C, B&C&D/BAR/E valid plans? As implemented here, the former *is* a valid plan, while the latter is *not* a valid plan. Why? In the case of the former plan, it is clear that the first schema expects that both B and C should be asserted after FOO happens; if this is in fact the case, then executing BAR (which only requires B) is fine; therefore, these two schemas chain. On the other hand, in the case of the latter plan, the first schema only predicts that two items should be asserted if action FOO is taken, yet the second schema will not be applicable unless an additional item (D) is asserted that the first schema did not predict. Since this is unlikely, these two schemas do *not* chain.

we did not somehow penalize paths for excessive length, then many paths might contain such "no-operation" schemas, since clearly inserting such a schema can be done in any case in which A is true.

Given all this, how are we to choose a reasonable path? We implement a *path metric* which, given a path, computes its merit; we then use the path which has the highest merit. (This path consists of all schemas whose actions we should take; it does *not* include the special "last" schema, as described above in Section 4.3.2.4.1, on page 74.) This is surprisingly tricky to do right; certain apparently-reasonable path metrics lead to grossly unstable planning behavior. Let us examine some of these cases.

- *Case 1.* The most obvious path metric is to simply multiply all the reliabilities together of the individual schemas making up the path. In other words, if R_n is the reliability of some schema n in a path, then the merit M of the entire path is simply $M = \prod_n R_n$.

  This simple metric leads to unfortunate consequences. Since all schemas have reliabilities strictly less than one, it tends to favor shorter paths, which appears reasonable on the surface. But, since the only thing the metric notices is reliability, what this *really* means is that it favors very short paths with very reliable schemas—with no concern for whether the individual schemas in the path say anything that actually helps get to the end goal. In practice, the paths are almost useless: the cost of adding even one additional schema to some path is so high that it is never worthwhile to do so—even if adding some schema is exactly what would make it more likely to reach the end goal.
- *Case 2.* The major problem with Case 1 above is that the paths were insufficiently predictive of the world's behavior. An easy fix is to make the merit

7. For example, in the infant/eyehand world, there are many reliable schemas of the form HP02/HANDF/HP02, which says, "If the hand is already fully forward, trying to move it forward will not do anything."

of any individual schema proportional to both its reliability R and to the number of items it mentions in its context (I_C) and result (I_R). Thus, for two schemas, both of *cardinality* $I_T = I_C + I_R$, and reliability R, we should favor the one with a higher value of I_T.

Naively applying this approach, by making a schema's merit $M = R \cdot I_T$, leads to disaster. In general, schemas mention at least one context item and one result item (the exception being schemas with null contents); thus, typical schemas have $I_T \geq 2$. This implies that M for a typical schema is greater than unity; this tends to exert a strong bias in favor of longer paths, as long as each step in the path contains a schema whose $I_T \geq 2$. The resulting paths are highly redundant, consisting of many actions followed at some point by their inverses, and the sheer length of the paths, even ignoring its redundancy, makes them quite unlikely to ever succeed.

- *Case 3.* The next obvious refinement to Case 2 above is to *normalize* the cardinality of each schema before computing its merit by dividing the raw cardinality I_T by the total number of sense bits that *could* be on in any schema. If this latter number is I_{TOTAL}, this new approach specifies $M = R \cdot \frac{I_T}{2 \cdot I_{TOTAL}}$. (We have to double I_{TOTAL} in the denominator, since a schema that made a prediction about every item in both its context and result would have a cardinality of $2 \cdot I_{TOTAL}$.)

Alas, this approach fails also, for the opposite reason as Case 2. By normalizing in this fashion, we exert a *very* strong selective pressure for short paths—because no schema has more than a few percent of all items specified, every schema added to the path beats down the total merit by approximately two orders of magnitude. Given that, no reliability, no matter how

good, can hope to beat the negative effects of lengthening the path by even one schema, so we tend to go for the shortest possible path (e.g., one action, i.e., two schemas, of which we do not count the last one). Among all possible one-action paths, we then pick the one with the biggest individual merit, which might favor a relatively unreliable schema (which should thus be of low merit) that just happens to have a large number of items. This was seen immediately in a run in the infant/eyehand scenario, wherein we picked a path of /EYEL/(VF20&VF21) (a schema with a null context), which, not unsurprisingly considering its lack of context, had a reliability of only $3.47 \cdot 10^{14}$: ridiculously small, but probably the best individual merit of the applicable schemas (e.g., those whose result mentioned some item we need to be on for the goal to be satisfied)—of which most probably only had *one* item in their result, so they lost.

- *Case 4.* Rather than normalizing merit by multiplying reliability by cardinality, this approach *exponentiates* reliability to the cardinality, e.g., $M = R^{I_T}$. It tends to push relatively reliable schemas that mention several items up near, but certainly never above, unity, while very quickly beating relatively unreliable schemas into the dust (and the more items they mention, the worse such unreliable schemas will fare). Note carefully that this is *not* the *normalized* cardinality defined in Case 3 above. That would basically yield the same answer as Case 3 itself did (and *did*, when it was accidentally written that way at first), because we would be exponentiating to tiny powers (like 10^{-14} or whatever), rather than to small positive integer powers instead (like 2 or 4).

The path metric actually used for the results presented here was therefore that in Case 4 above.

4.3.2.5 *Exceptional conditions while planning*

4.3.2.5.1 *Actions having "no effect"*

The planning situation is slightly more complicated than described above. At each step of plan execution, we check to see if the previous step led to no change in the perceived state of the external world, e.g., no changed primitive items. If so, we assume that the action *had no effect* and therefore that the plan *would not succeed* even if we ran to completion. (The most common case in which this is true is when attempting to take an action that would move the eye or the hand out of bounds, e.g., attempting to move the eye leftward when it is already as far left as it can go, etc; in the infant/eyehand world, for instance, if one uses a world exactly the size described in Section 2.2.1, on page 22, and in [Drescher 91], moving the eye at random will encounter a limit stop and result in no effect exactly 1/3 of the time.) While it is *possible* that a perfectly valid action might have no effect some of the time, and therefore it should simply be tried again if nothing seems to change, it is unlikely in the microworlds used here, and we must in any event decide how many times to try again before concluding that the action will *never* have the predicted effect; in this case, our threshold for making such a decision is one trial.

4.3.2.5.2 *Serendipity*

In addition to checking for cases in which the agent's actions seem to have no effect, we also check for *serendipitous completion* of the goal. In the simple microworlds used here, the chances for reaching the goal state are nontrivial; for very simple goals (e.g., get *anything* into *any* of the five coarse foveal visual regions, etc), the chances can be 10-20% that taking any of the actions permitted by the goal *at random* might nonetheless lead to a goal state. (For more complex goals, or in more complex microworlds, the chances of serendipitous completion can be arbitrarily low, of course—assuming that we start far enough from the goal state; see Section 4.3.2.7, on page 83, for some comments on that.)

Therefore, at each step in plan execution, we check for a serendipitous outcome, defined as reaching the goal state when there are still actions awaiting execution. Such a serendipitous result is not considered a planning success, since if it was because of some feature of the microworld (e.g., given enough knowledge, one might have predicted it), then it clearly indicates missing knowledge that should have been learned (and would have resulted in a shorter generated plan).

4.3.2.5.3 Goal loops

If incremental replanning is enabled during some run (meaning that the agent plans the entire path to the goal anew each time it takes an action), there is a possibility of *goal loops*. Such loops have been observed, and take the form of (say) a plan saying A/FOO/B, B/BAR/C, C/BAZ/D on some particular iteration. After taking action FOO, we then replan, and happen to get (say) B/BIFF/A, A/BOO/D, D/BAZ/C. We take action BIFF, replan, and end up with a new plan of A/FOO/B, B/BAR/C, C/BAZ/D again. (One particularly popular loop seen during one run involved a period-four loop (e.g., it took four actions to repeat itself), with two two-action plans and two three-action plans involved in the loop.)

In order to avoid this sort of pathological behavior, the number of times that replanning happens when pursuing a particular single goal from a starting state is tracked; if this number goes above a threshold, we assume that the large number of replans indicates that we are in a loop, and the current attempt to reach the goal is declared a failure. While it would certainly be possible to keep track of every single plan generated while pursuing some attempt at a goal, and immediately declare that we are in a loop if a repeat is seen, the approach taken has the features of simplicity and, perhaps, robustness—if something went wrong while executing a plan due to some unpredictable event (such as a random motion of an object outside of our control), and the replanned path happens to be the same as some previous path in this attempt, we should not gratuitously abandon the attempt. Simply

counting replans and aborting if the count exceeds a threshold works well in practice and is quite easy to implement, especially considering that planning loops are relatively rare anyway—though quite destructive if not caught.

This approach to avoiding loops has an intuitive feel like that of avoiding boredom—if some series of actions seems highly repetitious, we have been doing them for a long time, and the goal still has not been reached, perhaps it is time to give up on the attempt.

4.3.2.5.4 Other ways to be stymied

Many other things can go wrong while attempting to plan a path to a goal. This section details a few of the ways which we track explicitly. In all such cases, such a planning failure results in the complete abandonment of this attempt to reach the goal; we do a *relaxation* cycle to recover (see Section 4.3.2.7, on page 83).

When planning a path to a goal, we attempt to compute a path from some schema in the set INITIAL (consisting of all schemas whose contexts are currently satisfied by the currently-sensed state of the world) to some schema in the set FINAL (consisting of all schemas whose contents *would* be satisfied if the goal were to be satisfied). It may be that either one of these sets might be empty; this could happen if we know so few schemas (or so few schemas relevant to the current goal) that we cannot find even one such schema that could be satisfied in either the start or goal states. Such "stymied" configurations are referred to as *stymied-initial* or *stymied-final*, respectively.

It may also be the case that, even though INITIAL and FINAL are both nonempty, we still cannot find any path, no matter how bad, between at least one schema in each set. Such a failure is a *stymied-can't-start* failure, in which we cannot even start the planning process. A similar failure can occur if incremental replanning is allowed, in which we suddenly find, during a replan, that there is no path from the schemas in the *current* INITIAL

set (remember, this set will change with each change to the microworld) to FINAL. Such a failure is a *stymied-can't-continue* failure.

It may also be that the current state of the microworld, combined with the current goal, is such that some schema is in *both* INITIAL and FINAL simultaneously. This would indicate that, given the way the goal is specified, we need take *no* action to reach the goal—we are already in it. Such a situation is called a *short-circuit.*

This is distinct from a situation in which no element appears in both INITIAL and FINAL, yet the current state of the world is such that the context of some schema in FINAL is already satisfied. Such a situation is a *done-at-start* situation; we just happened to decide to reach a goal that we are already at.

4.3.2.6 Planning for learning versus planning for evaluation

As described above, the way that goals are used for learning are slightly different from the way that goals are used for evaluation of schemas. In particular, when learning, we run an FSM to determine which goal to execute after the current goal. In evaluation, we pick a goal in advance, and build a chain composed of schemas we may find in the memory to attempt to plan from the starting state to the goal state.

There is nothing in particular which stops us from running an FSM during evaluation as well; such a mechanism would constrain which schemas might possibly participate in a plan in the same way that the goal mechanism constrains which sensory items and schemas may participate in learning. However, since we are interested here in evaluating how much has already been learned by some prior run of the schema system, it seemed more appropriate to allow *all possible* schemas that have been learned to participate in path planning (though see Section 4.3.2.8, on page 84, for an exception to this).

4.3.2.7 Randomizing the world

It is very often the case that the current state of the world is unsuitable for performing an evaluation of the schema system's knowledge base.[8] Most of the reasons have been mentioned above; for example, being stymied in some way (see Section 4.3.2.5.4, on page 81) is the direct result of an interaction of the current goal and the current state of the microworld.

In most cases of being stymied, immediate abandonment of the plan is indicated. These cases are *stymied-short-circuit, stymied-can't-start, stymied-can't-continue, stymied-initial*, and *stymied-final*. In these cases, we *take aimless steps* by taking a fixed number of actions totally at random. By taking random actions, we side-effect the world in various ways; the world also has a chance to allow any other events that may happen independent of our actions to happen as well. The end result of this "aimlessness" is to do a random-walk away from whatever point in state-space caused us to be stymied. After this random walk, it makes sense to try the evaluation again on the current goal. If the agent repeatedly tries to evaluate given some goal, and repeatedly has to aimlessly wander away due to a planning failure, eventually it passes a threshold (denoted below as being *frustrated*) and abandons the goal, entering a relaxation cycle as described immediately below.

Other cases indicate abandonment of the goal itself. Not all of these cases are necessarily planning failures. If the agent is stymied because some action had no effect (e.g., *stymied-no-effect*), or if it succeeded serendipitously, or if it was frustrated by having to do too much aimless wandering, these *are* planning failures; however, successful completion of the goal at the predicted instant (e.g., at the end of the plan) is a planning *success*. In these cases, the agent *relaxes*, also by taking random actions, in an attempt to random-walk away

8. This is independent of the state of the knowledge base. For instance, if we have just succeeded in getting to some end goal, and then try to run an evaluation to that same end goal, we surely should not start from where we left off, because it would take zero steps to get to the goal. This would tell us nothing.

from the current state; this is identical to "aimlessness" as described above, though the number of steps taken during relaxation may or may not be the same (in the current system, they happen to be the same). When done relaxing, the agent takes the another goal from the set of goals being used to evaluate the competence of the knowledge base, and starts over.

4.3.2.8 Lobotomies

It is often useful to evaluate, not only how one set of goals compares to another in producing useful learning for some task, but *how fast* that learning takes place, or after *how many facts are learned* that the agent may be said to be competent.

In general, this was done by generating a full knowledge base and then *lobotomizing* it by making some percentage of it invisible to the evaluation system, simulating earlier states of knowledge.

4.3.2.9 Scorecards

Given all the above, an overview of the evaluation process is straightforward. For any given evaluation, we must decide the conditions under which the evaluation takes place, namely:

- the knowledge base of schemas to be evaluated
- which goals will be used
- the extent of any lobotomies
- whether or not incremental replanning is allowed

Once these parameters are chosen, the procedure for evaluating a run is to repeatedly attempt to reach a goal from the starting state, as described above. If multiple goals are specified, we *round-robin* among them; this makes it more likely, even with aimless or relaxation, that the next goal to be evaluated will not have the state of the world already preset to a goal-satisfied configuration. Very often, runs will include various cross-product

settings of the parameters (e.g., trying several goals, with and without incremental replanning, over some set of lobotomy values).

Such simulation runs are organized into a system of *scorecards;* each scorecard is keyed by the name of some goal. A scorecard is essentially a bag of state, reflecting many of the quantities above, plus a few new ones. The complete list is:

- *N*: Number of times the agent has tried with this goal.
- *Goal*: The goal the agent is trying to achieve.
- *Wins*: How many times the agent was able to plan *and* reach the goal; does *not* include serendipity or aimless-wins.
- *Stymied-initial*: How many times the agent was not even able to begin planning (INITIAL empty).
- *Stymied-final*: How many times the agent was not even able to begin planning (FINAL empty).
- *Stymied-can't-start*: How many times the agent could not compute an initial plan.
- *Stymied-can't-continue*: How many times the agent was suddenly unable to continue after a replan (no new plan can be formed).
- *Stymied-short-circuit*: How many times the agent was suddenly short-circuited.
- *Stymied-no-effect*: How many times no primitive items were changed by the action the agent just took.
- *Serendipity*: How many times the agent succeeded earlier than the currently-amended plan thought it would.
- *Done-at-start*: How many times the agent was done before taking any action at all.

- *Replans*: How many times the current plan changed (ignoring simple shortening as it is executed).
- *Replan-loops*: How many times the agent exceeded the maximum permissible number of replans when trying to collect a card.
- *Aimless-steps*: Number of aimless steps the agent took while hoping to encounter a state it could plan from.
- *Totally-frustrated*: Number of times the agent gave up after being aimless too long.
- *Ave-winning-plan-length*: Average length of any winning plan or replan (from when it was initially planned, not at the end, of course).
- *Ave-serendipitous-plan-length*: Average length of the remaining length of the plan when it was discovered that the plan was serendipitously done.
- *Contextless-path-starts*: Number of times the agent picked a schema with an empty context as the start of the best path.

The results presented below make use of this terminology and organization of scorecards.

4.4 Results

4.4.1 The work required for learning with goals

Goals can exert a powerful effect on the work required to learn. Figure 10, on page 88, shows that a run with goals as described above in Figure 9 leads to substantially less work than the goal-independent case, for similar numbers of schemas. For comparison purposes, this chart uses the same microworld as the chart shown in Chapter 3, which was run in the infant/eyehand scenario.[9]

This is quite similar to the sort of results obtained in goal-independent learning. After all, in the model used in this research, the imposition of goals can only decrease the number of percepts which are attended to, either by the sensory or the cognitive systems. Hence they will tend to act as selectors or filters, just as in the goal-independent case. The effect of the restriction of possible actions is less clear in advance; as the table demonstrates, however, adding goals in fact serves to greatly increase the efficiency, in terms of computational work.

However, this is not the whole story. After all, as explained at great length in Section 4.3.2, on page 71, goals can dramatically and qualitatively change the nature of what facts are learned about the world, by directing the learning effort away from particular features and toward others. The following sections will investigate those effects.

4.4.2 Changes in the characteristics of the schemas learned

One of the most obvious effects of learning with a particular goal in place is that very few schemas addressing parts of the world that are not deemed *relevant* by the goal are generated. Consider the goal shown in Figure 9, on page 69. This goal is concerned with building competence at centering objects in the visual field and identifying them. As such, it does not attempt to manipulate the objects. Because this is so, the goal never allows the hand to move, which means that the agent will never discover what would happen if it *did* move the hand. (This is not completely true, because of the occasional randomness discussed in Figure 4.3.1.1, on page 64—but it is virtually true.)

9. Some analogous goal runs have been accomplished in the Hamsterdam scenario. For similar goals (e.g., center an object in the sensor fan), similar performance can be obtained, which is unsurprising (though reassuring) considering the approximate correspondence of actions and sensor systems between the two scenarios. The Hamsterdam scenario also offers the possibility of creating goals that depend on dynamic aspects of the environment (for instance, learning to chase a moving object), however, work on such more-sophisticated goals is still preliminary.

Algorithm				Learning			Work required			Facts per work unit		
Spinoff selectors		Statistic selectors		Schemas			Inner loops (x10^6)			Reliable schemas over		
Items	Schemas	Items	Schema	Total	Rel	T/R	Spin	Stat	Both	Spin	Stats	Both
AIN	**ASN**	**AIN**	**ASN**	**1756**	**993**	**1.77**	**533**	**533**	**1066**	**1.9**	**1.9**	**0.9**
AIN	ASN	CINIH	ABSPSDUCI	1135	403	2.82	398	12	410	1.0	33.6	**1.0**
AIN	ASN	AIN	ABSPSDUCI	1110	518	2.14	391	55	446	1.3	9.4	**1.2**
CIN	**ASN**	**AIN**	**ASN**	**1693**	**948**	**1.79**	**44**	**524**	**568**	**21.5**	**1.8**	**1.7**
CIN	**SWRUS**	**AIN**	**ASN**	**1395**	**791**	**1.76**	**2**	**463**	**466**	**316.4**	**1.7**	**1.7**
CIN	ABSPSDUCI	AIN	ASN	1622	924	1.76	15	510	525	61.6	1.8	**1.8**
CIN	ASN	AIN	ABSPSDUCI	1110	506	2.19	33	54	87	15.3	9.4	**5.8**
CIN	ABSPSDUCI	AIN	ABSPSDUCI	1110	506	2.19	10	54	64	50.6	9.4	**7.9**
CIN	ABSPSDUCI	CINIH	ASN	1366	643	2.12	13	64	77	49.5	10.0	**8.4**
CIN	ASN	CINIH	ABSPSDUCI	1136	399	2.85	34	12	46	11.7	33.3	**8.7**
CIN	**SWRUS**	**AIN**	**ABSPSDUCI**	**1102**	**498**	**2.21**	**1**	**53**	**54**	**415.0**	**9.4**	**9.2**
CIN	SWRUS	CINIH	ASN	1353	688	1.97	2	64	66	275.2	10.8	**10.3**
CIN	ABSPSDUCI	CINIH	ABSPSDUCI	1136	399	2.85	10	12	22	40.7	33.3	**18.3**
CIN	**SWRUS**	**CINIH**	**ABSPSDUCI**	**1134**	**398**	**2.85**	**1**	**12**	**13**	**331.7**	**33.2**	**30.2**
CIN	**SWRUS**	**CINIH**	**ASN**	**461**	**229**	**2.01**	**.38**	**7**	**7.4**	**605**	**31.9**	**30.3**
CIN	**SWRUS**	**CINIH**	**ASN**	**336**	**178**	**1.89**	**.38**	**6.4**	**6.8**	**475**	**27.7**	**26.1**
CIN	**SWRUS**	**CINIH**	**ABSPSDUCI**	**227**	**111**	**2.05**	**.17**	**1.9**	**2.1**	**634**	**57.2**	**52.4**

Figure 10: Goal-dependent learning and its effect on the work required

This table is a recap of Figure 8, on page 55, with additional results for a typical goal run (in this case, the goal demonstrated in Figure 9, on page 69). The new results are in the box at the bottom. Goal-independent selectors ***and*** *the goal mechanism are run* ***simultaneously****.*

The first, lightly shaded line shows a run with the given selectors and the goal of Figure 9, in which the occasional randomness in permitted actions described in Section 4.3.1.1, on page 64 is disabled (in other words, the permitted actions are always completely determined by the specification of the active goals).

The remaining pair of more heavily-shaded lines show results in which occasional randomness in permitted actions is enabled (the normal case). The top line of the pair is exactly the same run as that shown by the lightly-shaded line above it. The bottom line of the pair shows the "best" strategy from Chapter 3 (on the bottommost line of that table, and immediately above the box in this one) combined with the goal from Figure 9.

A less obvious effect concerns the *coverage* of that part of the world that is deemed important by the goal (in other words, how thoroughly the generated schemas make predictions about all the possible states of the world and the effects of the agent's actions in those states). Consider conjunctions, which are used whenever a schema's context or result must talk about more than one item, and synthetic items, which are used to provide the basis for the mechanism used to learn concepts such as object persistence [Drescher 91].

The table below compares the number of generated conjunctions and synthetic items

Algorithm				Goal	Learning			
Spinoff selectors		Statistic selectors			Schemas		Items	
Items	Schemas	Items	Schema		Total	Rel	Conj	Syn
AIN	ASN	AIN	ASN	none	3244	1856	184	10
CIN	SWRUS	CINIH	ABSPSDUCI	none	3213	1299	112	13
CIN	SWRUS	CINIH	ABSPSDUCI	High Res Scan	3209	1317	994	119

Figure 11: The effect of goals on selected schema characteristics

generated in goal-independent and goal-dependent learning. First, two goal-independent runs, which were created using different selectors, are compared. It is clear that using a more-focused algorithm tends to decrease the number of conjunctions learned, though only by a small amount. However, adding goal-dependent learning dramatically increases the number of conjunctive contexts and results in the generated schemas, by a factor of 5.4 over the least-focused algorithm (in which we are comparing apples and oranges, really–we are comparing unfocused, goal-independent learning with focused, goal-dependent learning), and by a factor of 8.9 when the selection algorithm is held constant. (In other words, if we compare two runs which were both generated with the same type of goal-independent filtering—as shown by the bottom two lines of the table—and then use goal-dependent learning in one of those runs, the number of conjunctions in the run that used goal-dependent learning is vastly increased.) These figures are for a particular number of generated schemas, of course: these ratios will change as the length of the run changes. In

general the number of generated schemas, conjunctions, and synthetic items all follow an n^2-shaped curve as the number of iterations increases, starting off slowly and then growing more and more quickly.

Similarly, the number of synthetic items generated has also increased enormously, by about an order of magnitude.

What can be causing these changes? It appears that, by restricting the learning to schemas making predictions about only some aspects of the given microworld, we are decreasing the *breadth* of learning, while increasing its *depth*. Certain things are learned much more slowly (e.g., in this case, the effects of hand motions), but those things which *are* learned (e.g., the effects of eye motions) are learned in much greater detail.[10]

To demonstrate the above claim about increased depth, let us examine the average cardinality of the generated schemas. (Cardinality was defined in Section 4.3.2.4.2, on page 75, in the discussion of path metrics. It is equal to the number of items appearing in the context and result of a particular schema.) If conjunctions are more common, the average cardinality is higher, since each schema mentions more items on average.

The table below shows that, for roughly comparable numbers of generated schemas,

Algorithm				Goal	Learning			
Spinoff selectors		Statistic selectors			Schemas	Ave Cardinality		
Items	Schemas	Items	Schema		Total	Ctxt	Res	Both
AIN	ASN	AIN	ASN	none	3599	0.99	1.15	2.14
CIN	SWRUS	CINIH	ABSPSDUCI	High Res Scan	3366	1.72	1.32	3.04

Figure 12: Average schema cardinalities with and without goals

the average cardinality of schemas has increased substantially–each schema, on average,

10. If one had the time to run a much larger number of iterations, as Drescher has done on various types of Connection Machines, one might see these large numbers of conjunctions and synthetic items even if no focus is being employed, since the entire microworld will be very thoroughly covered if enough schemas are generated. Indeed, this is exactly the case, as reported for long runs on parallel machines, producing tens of thousands of schemas, in which conjunctions and synthetic items dominate the results [Drescher 93].

makes a prediction about more of the world.[11] The first line shows a run that uses the original, unfocused learning algorithm—which tends to generate more conjunctions than any of the more-focused algorithms—when running in a goal-independent fashion. The second line shows the most tightly-focused algorithm—which would normally generate many fewer conjunctions than the unfocused algorithm, if it were running goal-independently—in which the algorithm is being run with goal-dependent learning. Even with the odds stacked against the tightly-focused run on the second line, the addition of goal-dependent learning clearly shows that the average cardinality has increased.

4.4.3 Competence

As examined in Section 4.3.2, on page 71, simple comparisons of the numbers of schemas generated is not necessarily sufficient when reasoning about the effects of learning with goals. In this section, we use the terminology and methods of that section to evaluate what the schema system has learned when operating in a goal-directed fashion. To keep the length of the presentation reasonable, we shall again use the goal shown in Figure 9, on page 69, as our exemplar.[12]

11. There are two possible confounding influences in this chart, but both are relatively minor. First, we are making a comparison with different selectors. However, Figure 11, on page 89, demonstrates (as do other results not shown here) that using a focused, goal-independent run, instead of the unfocused one used here, would tend to generate somewhat fewer conjunctions for a given number of schemas, hence resulting in lower average cardinalities–which would only increase the contrast seen between the goal-independent and goal-dependent results presented. Second, we are assuming, when we say that "the goal-dependent schemas are predicting more about the world," that the average reliabilities of schemas in each set is comparable.This is in fact the case.

12. Incidentally, several other goal sets have been specified and run, although many of them are less interesting than the one used repeatedly here. For example, a goal set which attempts only to get any object into any foveal region was tried early on—but its performance is uninteresting, because almost any eye motion will tend to get an object into some portion of the visual field, in either scenario, because of the relatively larger percentage of the eye occupied by the fovea. Other, more difficult goal sets showed similar properties to those related here for object centering.

In general, the learning was evaluated by generating a set of scorecards for any given combination of parameters. Usually, the scorecard set is composed of simulation runs with different lobotomy levels imposed, to examine how learning longer affects the results.

A few scorecard sets from the larger number[13] generated for various goal sets are shown in Figure 13, on page 93, and Figure 14, on page 94.[14] When examining these scorecards, which report (among many other things) the average length of plans to reach the end goal, it is reasonable to ask how long a path a *random* plan might be expected to produce, in other words, the length of a random walk from a random point in the state space to the goal state.

For the goal shown in these charts, the length of such a random plan is approximately 13.4 steps. This was determined by the simple expedient of taking a random action at every timestep, and counting up the length of each plan from one encounter of the goal state to the next, over several hundred thousand timesteps.

Figure 13 demonstrates the effects of grossly insufficient learning, caused by lobotomizing the knowledge base at a very small number of schemas. Until we get to somewhere above 90 schemas in this case, there is no schema which actually includes the goal state in its result. Before this point, planning cannot even begin, as no chain may be built from *any* initial state to the goal state. The number of aimless steps taken while attempting to recover from these planning failures is quite large; so is the number of times that the pursuit of the

13. On the order of a couple of dozen, for various goal sets, including scanning into anywhere in the fovea, scanning into dead center, moving the hand until it contacted something, scanning to a particular spot on the edge of the visual field (inspired by orientation to peripheral vision), and so forth.

14. *N* in these charts shows the number of attempts made to plan a path to the goal; we run any given scorecard either until 50 wins have occurred, or until we have tried 100 times to begin planning. Thus, if a large number of events which prohibit even trying to plan occur in a run (for example, many instances of *done-at-start* or *stymied-short-circuit* failures, in which we cannot even being planning until the world state has changed somewhat), *N* will be substantially less than 100.

High Res Scan Into Dead Center (goal-INdependent, incremental planning ENabled)																	
Lobot		Expected		Unexpected				Stymied						Despr	Relaxation		World
Limit	N	Wins	WPLen	Seren	SPLen	Repln	RpLen	Init	Final	¬Start	¬Cont	¬Effct	ShtCkt	¬Cntxt	Aimless	Frust	DoneS
370	77	13	1.00	1	2.00	3	0	0	0	0	0	9	0	41	0	0	27
270	65	12	1.00	0	0.00	0	0	0	0	0	0	11	2	50	0	0	15
240	67	11	1.00	0	0.00	0	0	0	0	0	0	8	0	50	0	0	17
210	69	9	1.00	0	0.00	0	0	0	0	0	0	9	0	50	0	0	19
180	63	11	1.00	0	0.00	0	0	0	0	0	0	14	0	50	0	0	13
150	60	16	1.00	0	0.00	0	0	0	0	0	0	2	0	50	0	0	10
120	63	13	1.00	0	0.00	0	0	0	0	0	0	11	0	50	0	0	13
90	63	0	0.00	0	0.00	0	0	0	0	50	0	0	0	50	521	13	13
60	67	0	0.00	0	0.00	0	0	0	0	50	0	0	0	50	587	17	17
30	67	0	0.00	0	0.00	0	0	0	0	50	0	0	0	50	544	9	17

Figure 13: A typical scorecard set for a short, goal-independent run

This table shows a run with a very small knowledge base of schemas, generated during goal-independent learning of the type described in Chapter 3. Further discussion in the text; note in particular the large number of "can't-start" planning failures until a schema mentioning the goal state exists.

"N" is explained in the text. "Limit" shows the number of "visible" schemas under the current lobotomy. "WPLen" and "SPLen" show the average length of winning and serendipitous plans, respectively; "PLoop" shows the number of planning loops detected (see Section 4.3.2.5.3, on page 80). These and the other table headings correspond to those described in the discussion of scorecards in Section 4.3.2.9, on page 84.

High Res Scan Into Dead Center (goal-DEpendent, incremental planning DISabled)																	
Lobot		Expected		Unexpected				Stymied						Despr	Relaxation		World
Limit	N	Wins	WPLen	Seren	SPLen	Repln	PLoop	Init	Final	¬Start	¬Cont	¬Effct	ShtCkt	¬Cntxt	Aimless	Frust	DoneS
3367	60	46	1.61	2	2.00	0	0	0	0	0	0	0	0	2	0	0	10
3030	59	39	1.59	6	2.00	0	0	0	0	0	0	2	0	4	0	0	9
2693	68	47	1.79	1	2.00	0	0	0	0	0	0	0	0	0	0	0	18
2356	65	47	1.96	1	2.00	0	0	0	0	0	0	1	0	0	0	0	15
2020	61	40	1.77	3	2.00	0	0	0	0	0	0	3	0	4	0	0	11
1683	67	35	2.20	7	2.57	0	0	0	0	0	0	5	0	7	0	0	17
1346	71	27	1.52	2	2.50	0	0	0	0	0	0	10	0	18	0	0	21
1010	65	20	1.80	2	3.00	0	0	0	0	0	0	13	0	28	0	0	15
673	59	21	1.86	19	3.00	0	0	0	0	0	0	9	0	11	0	0	9
336	75	19	1.68	8	2.50	0	0	0	0	0	0	11	0	21	0	0	23

High Res Scan Into Dead Center (goal-DEpendent, incremental planning ENabled)																	
Lobot		Expected		Unexpected				Stymied						Despr	Relaxation		World
Limit	N	Wins	WPLen	Seren	SPLen	Repln	PLoop	Init	Final	¬Start	¬Cont	¬Effct	ShtCkt	¬Cntxt	Aimless	Frust	DoneS
3367	75	44	1.14	5	2.20	28	0	0	0	0	0	1	0	2	0	0	23
3030	68	47	1.17	1	3.00	24	0	0	0	0	0	0	2	0	0	0	18
2693	63	49	1.12	0	0.00	29	0	0	0	0	0	0	0	1	0	0	13
2356	63	43	1.12	3	2.33	26	0	0	0	0	0	1	0	2	0	0	20
2020	70	39	1.00	10	2.40	57	0	0	0	0	0	1	0	2	0	0	20
1683	60	39	1.00	8	2.00	37	0	0	0	0	0	1	0	6	0	0	10
1346	65	26	1.65	9	2.78	22	0	0	0	0	0	8	0	35	0	0	15
1010	60	24	1.00	6	2.33	25	0	0	0	0	0	10	0	33	0	0	10
673	68	35	1.66	12	3.83	35	0	0	0	0	0	3	0	15	0	0	18
336	70	27	1.48	3	3.00	16	0	0	0	0	0	11	0	32	0	0	20

Figure 14: A couple typical scorecard sets for some long, goal-dependent runs

goal is declared to be totally frustrated (because the simulation *never* manages to take enough aimless steps that it lands in a state from which planning may proceed).

Even after we pass the point at which there are enough schemas that planning can commence, the results are poor with this few schemas. There are few wins and many types of planning failures, such as no-effect stymies. Almost every plan starts with a schema with a null context (a desperation plan). The average winning path length is 1.00—because any plan involving more than one step ends in failure due to insufficient knowledge about the world, and hence is not counted in an average of *winning* path lengths.[15]

Figure 14, on the other hand, compares two otherwise-similar runs with a much larger number of schemas. One run allows incremental replanning (bottom table), and one does not (top table). Allowing replanning while attempting to reach the goal demonstrates more clearly that the schemas have learned what to do: the average path length to the goal is significantly shorter when replanning is allowed, since incorrect predictions or changes in the world can be compensated for en route. When the knowledge base has been extensively lobotomized (e.g., below around 1300 schemas in this example), replanning tends to lose its effectiveness, however. This can be seen by several factors, such as the high average plan length, even for winning plans; the number of times an action has no effect; and the number of times that a plan that starts with a contextless schema, indicating a desperation move. In short, when very little is known about the domain, planning more frequently is of no use.

As a final exercise, let us compare goal-independent with goal-dependent competence directly, using as parameters the number of path wins, the number of "no effect" stymies,

15. Indeed, it is quite likely that almost all of those one-step plans were desperation plans, which also explains their low reliability. If the plan just happened to start from a state that was one step away from the goal (though it did not really "know" it, because its context was empty), then it might win. But if it was any farther away, the lack of a context at its start dooms the plan to failure, because it essentially "looked before it leaped."

and the number of "no starting context" (desperation plans). Figure 15 below examines

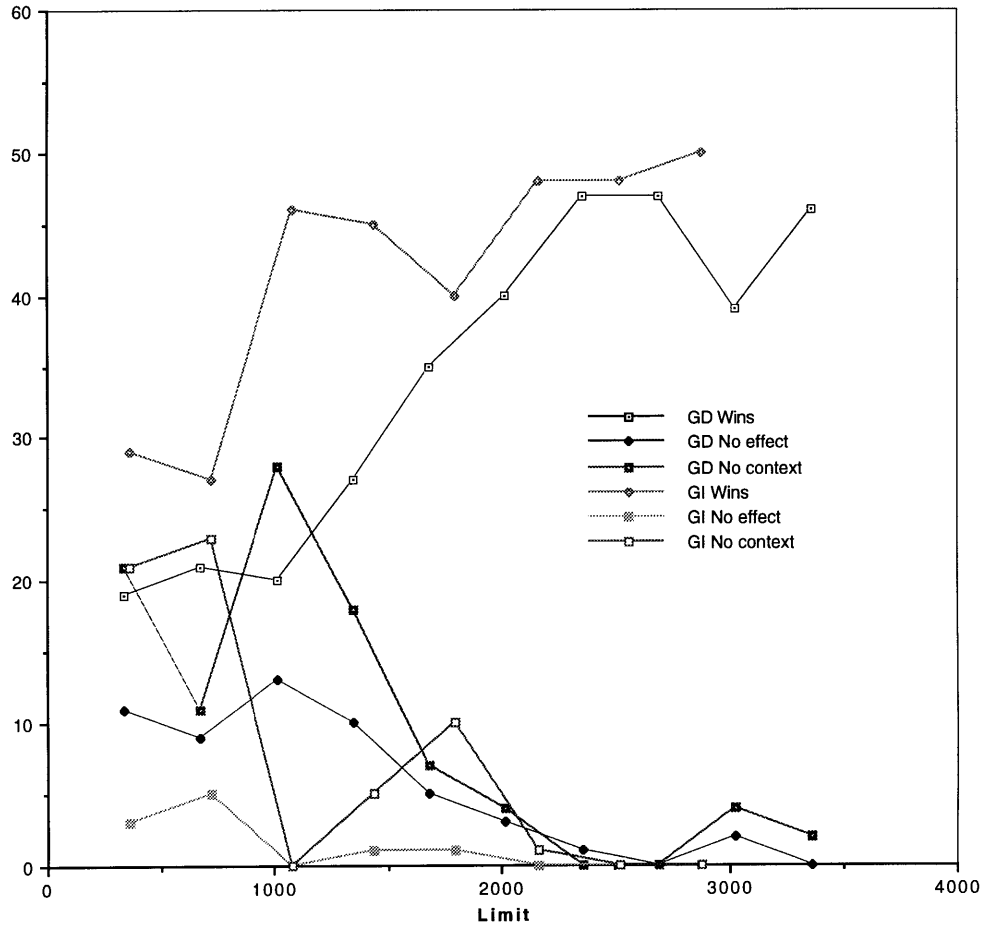

Figure 15: Goal-independent and goal-dependent performance compared

these variables. It shows the number of schemas in the knowledge base increasing towards the right ("limit" shows the top schema number allowed in the range permitted under the current lobotomy and is equal to the number of schemas). The vertical axis is a simple count of the various types of planning scorecard entries for wins, no-effect stymies, and no-context (desperation) plans, for both goal-dependent and goal independent learning.

In general, and as might be expected, planning successes for either method increase as the number of schemas increases, and stymies and desperation plans decrease. (After all,

all graphs of learning algorithms show this sort of trend, since learning systems that become less competent as they learn more are generally considered uninteresting.)

The most peculiar thing about this graph is that goal-independent learning does so well. One might imagine that, having learned less deeply but more broadly, goal-independent planning would do less well than goal-dependent learning.[16] Instead, goal-independent learning does better, at least for our example goal set. What could explain this surprisingly good performance?

Part of the answer lies in the relative simplicity of the microworlds employed in either scenario, and in the goals we use. There are only a small number of unique objects in either microworld; this has interesting consequences. Consider the infant/eyehand scenario when using goal-independent learning. At any given point in a run, even if relatively few schemas mentioning coarse visual items have been generated (relative to all schemas which have been generated) there are a large number of schemas dependent upon fine foveal visual items, which make predictions about particular details seen in the fovea in response to moving the eye.[17] Since there are so few objects (e.g., four), the number of different configurations of perceivable details is small; this means that fine foveal items can essentially take the place of coarse visual items in predicting the result of eye motions.

16. Of course, all of this assumes that, if we are testing goal-dependent learning, the goals used in learning have something to do with the goals used in evaluation. If we were to train the system by only running goals involving eye motions, then evaluate the system by only running goals involving hand motions, we would rightly expect very poor performance. Indeed, if the "occasional randomness" factor for permitted actions is turned off, performance would be uniformly zero in this case.

17. In the infant/eyehand scenario, fine foveal items make up approximately half of *all* sensory items. Coarse visual items make up another quarter or so. This means that the contribution of haptic, taste, etc, inputs is small, so adding in schemas which also make predictions about those inputs (as the goal-independent case does) tends not to "dilute" the general pool of schemas with non-visually-predictive schemas. Hence, by being completely unselective, we do not materially affect the performance of *visual* goals in this microworld—*if and only if* such goals involve the region in or near the fovea in some way.

To be more specific, consider a plan such as VF42/EYER/VF32, VF32/EYER/VF22. The end result of this plan is to activate item VF22, e.g., to land something dead-center in the coarse visual field: such a plan should only be invoked if the agent started out with some object at VF42. Suppose, however, that VF32/EYER/VF32 did not exist as a reliable schema in the current knowledge base—this might well be the case, even in a large knowledge base, in a goal-independent run (see also the discussion a few paragraphs below and Figure 16, on page 100, for a real example of this).

A goal-independent run, though it was missing VF42/EYER/VF32, would be quite likely to have *some* schemas such as VF42/EYER/FOVR03 and FOVR03/EYER/VF32, for some foveal detail in FOVR (in this case, it happened to be 03).[18] In this case, we can nonetheless form the plan VF42/EYER/FOVR03, FOVR03/EYER/VF32, and we are quite likely (because there are so many foveal sensory items—roughly half of all items) that there will be some pair of schemas that can continue the chain in this fashion.

In essence, plans that would otherwise use coarse visual items may also use fine visual items, with equivalent predictive power, because large "clumps" of fine foveal items behave en masse, hopping as a group from one foveal region to another in the same way that single coarse visual bits do under similar actions.[19]

18. FOVR is the right foveal area; it covers exactly the same area as VF32.

19. Because so many foveal items hop "as a group" (there being only a very small number of distinct clumps of details, given the small number of objects), highly-conjunctive schemas are also common. In the goal-independent case, most of the conjunctions in schemas are conjuncts of multiple fine foveal items; in the goal-dependent case, most of them are conjuncts of coarse visual items, arrived at with considerably more experimentation due to their poorer correlations in the world. This means that, in the goal-independent case, the large number of correlated foveal bits tend to cause the Case 4 path metric (Section 4.3.2.4.2, on page 75) to preferentially use them as well. This large number of conjunctions means that we may find a large number of paths such as, e.g., VF42/EYER/FOVR01&FOVR02&FOVR03&FOVR04, FOVR03/EYER/VF32, since the second schema need only depend on *one* of the bits asserted by the first, and since any object that causes one foveal bit to be asserted will tend to cause many others in that foveal area to be asserted simultaneously as well, leading to a large number of generated conjunctions.

This outcome was unexpected. While goal-dependent learning clearly uses less computational work and still produces quite reasonable results, the ability of goal-independent learning to "compensate" by using, in this case, fine foveal items, makes it difficult to see the advantages of goal-dependence.

There are several ways of constructing better test cases for goal-dependent learning, given the situation. Results from one such test case are illustrated in Figure 16, on page 100.[20] This pair of scorecards shows performance for the somewhat-peculiar task of scanning an object into coarse visual field position 0,1—a point chosen completely at random on the periphery of the visual field, in this case near the lower left corner.[21]

Since there are no fine foveal inputs there, foveal schemas cannot help us. In this case, it could be expected that goal-independent learning would do substantially worse than goal-dependent learning. In fact, as Figure 16 demonstrates, goal-independent learning for this randomly-chosen goal was a complete disaster—even with as many schemas as existed in the entire goal-independent run, the knowledge base was apparently completely unable to determine how to achieve the goal.[22]

The reason for this is clear from the above discussion. Since the goal-independent case allocates its learning effort more indiscriminately, it covers any particular part of the state space less thoroughly. In this case, even after generating 3600 schemas, it never happened

20. Another solution, which received some experimental attention, but not enough to report here quantitatively, is to employ microworlds with larger "diameters" (e.g., in the infant/eyehand scenario, one might expand the world from 7-by-7 to 20-by-20 or more) or, given a larger diameter universe to work in, a sensory systems with similarly larger diameters (making the fovea cover proportionally less area in the eye). Such changes would make longer paths more important for correct planning, which would tend to expose even small weaknesses; the combination of a long path and a small fovea should make goal-independent learning rather poor.

21. Such a task is akin to a human trying to see a very dim object by scanning it to the retinal periphery, where the high-sensitivity rods reside.

22. It so happens that the particular goal-independent run used was one using the original (unfocused) Drescher algorithm—so if any goal-independent run should have picked up the necessary schemas, this one should have.

Low Res Scan to VF01 (goal-INpendent, incremental planning DISabled)																	
Lobot		Expected		Unexpected				Stymied						Despr	Relaxation		World
Limit	N	Wins	WPLen	Seren	SPLen	Repln	PLoop	Init	Final	¬Start	¬Cont	¬Effct	ShtCkt	¬Cntxt	Aimless	Frust	DoneS
3600	53	0	0.00	0	0.00	0	0	0	50	0	0	0	0	0	957	0	3
2400	52	0	0.00	0	0.00	0	0	0	50	0	0	0	0	0	976	0	2
1200	51	0	0.00	0	0.00	0	0	0	50	0	0	0	0	0	990	0	1

Low Res Scan to VF01 (goal-DEpendent, incremental planning DISabled)																	
Lobot		Expected		Unexpected				Stymied						Despr	Relaxation		World
Limit	N	Wins	WPLen	Seren	SPLen	Repln	PLoop	Init	Final	¬Start	¬Cont	¬Effct	ShtCkt	¬Cntxt	Aimless	Frust	DoneS
3367	53	7	1.57	1	2.00	0	0	0	0	0	0	10	0	39	0	0	3
2244	54	3	2.00	2	2.00	0	0	0	0	0	0	9	0	38	0	0	4
1122	51	1	1.00	0	0.00	0	0	0	0	0	0	8	0	25	0	0	1

Figure 16: Superior performance from goal-dependence

This demonstrates a complete planning failure of the goal-independent learning system when given a random non-foveal task. Since it did not cover the state space with sufficient depth, even given 3600 schemas, it was never able to start the planning process, and experienced a stymied-no-final failure every time it did not serendipitously start in the goal state.

In contrast, the goal-dependent learning system, even when trained with a goal that was only somewhat-related to the task at hand (the training goal was High Res Scan Into Dead Center, which is a visual-scanning task but one biased toward the fovea), was able to plan paths to the goal that increased in accuracy as the number of schemas increased.

to generate a schema whose result was VF01. This lead to a complete failure of the planning system, as illustrated by the fact that a *stymied-can't-start* failure occurred for every attempt to plan. The only other outcome of any attempt to reach the goal in this case was the infrequent random occurrence of the goal already being true by chance when the system was about to try planning a way there (the *done-at-start* values in the rightmost column).

On the other hand, the goal-directed case covered the state space much more densely. Even with a third as many schemas as the goal-independent case, it was never unable to generate a plan. As schemas are added to the knowledge base, the accuracy of the plans apparently increases (note the increasing number of wins). There are still a fair number of mis-steps and desperation plans, evidently caused by still-incomplete coverage of the state space, but at least the goal-dependent case makes some progress. It would probably make substantially better progress if it had been trained with the same goal—in fact, this run shows the performance in trying to reach a goal (VF01 asserted) for which the system was not even explicitly trained—its training consisted of the *High Res Scan Into Dead-Center* goal, which has a different emphasis in the state space it attempts to investigate (but one which is still more biased towards visual scanning behavior than a goal-independent strategy).

One of the most important lessons from these results is that there is often very tight coupling between the world, the sensory system, and the goals that might be employed. All three interact quite strongly, and untangling that interaction can require persistence and care.

Chapter 5: Related Work

Work in selective attention can draw from two major fields for inspiration and guidance. The first is work in machine learning, primarily that concerned with causal model builders and active agents, and secondarily passive learners and the reinforcement learning literature in general.

The second is the study of attentional processes in the cognitive science literature. There has been considerable work on attention in cognitive science, and the questions asked and insights gained into attentional processes, while themselves often insufficiently well-specified to serve as computational theories, may serve as inspiration for approaches to machine implementation.

5.1 Related Work in Machine Learning

5.1.1 Introduction

As illustrated earlier, a typical agent in the world cannot perceive every part of the world at once, nor should it—even "perceiving" without "learning" is expensive if the agent must perceive everything. However, not perceiving the whole world at once can lead to a phenomenon that [Whitehead and Ballard 90] calls *perceptual aliasing*, in which different world states can appear identical to the agent, and which causes most reinforcement learning mechanisms to perform poorly or not at all. Both they and [Woodfill and Zabih 90] propose systems which combine selective visual attention (which is used to

"ignore" certain parts of the world at certain times) with special algorithms to attempt to overcome the aliasing problem.

Many of the methods described in earlier chapters may be available to other machine learning systems, and several such extensions are discussed in later sections. While Drescher's schema system keeps exhaustive statistics and is thus easy to adapt in the manner shown, *any* agent that tries to correlate its actions with results must keep around *some* sort of statistics regarding those results from which to learn, even if they are only available for the instant of perception, and those stored statistics are candidates for pruning. Further, any such agent must somehow perceive the world, and its sensory inputs are likewise candidates for pruning.

For example, the techniques used in the most-focused of the goal-independent strategies shown in Chapter 3 (bottom line of Figure 8, on page 55) are likely to be available to most learning systems operating in a discrete microworld. They require being able to keep track of which sensory items have changed recently, and which facts depend upon (e.g., make predictions concerning) those items. This does not seem an insurmountable obstacle for many algorithms. It is even possible that particular algorithms which do not possess absolute knowledge about, for example, which sensory items are mentioned in any given learned fact (such as the hidden nodes of a neural net) might nonetheless be able to yield a probabilistic estimate of how likely it is that some particular part of the internal knowledge base might depend on a particular sensory input. If so, such algorithms might also allow cognitive pruning to take place.

Selective attention and goal-directed learning have recently been getting considerably more attention in the literature than previously. Several researchers have advanced frameworks or architectures for thinking about and taxonomizing such systems, usually based either upon a model of *filtering* or *discarding* information deemed unnecessary or harmful

for the learner [Markovitch and Scott 93], or upon explicitly modelling goals, using goals both to inform the learning process and as input to metareasoning strategies which reason about the performance of the learner [Ram and Leake 94].

The next few sections will discuss some of these issues. Section 5.1.2, examines the selective attention side of the issue, and Section 5.1.3, on page 108, addresses goal-directed learning aspects.

5.1.2 Selective attention as filtering

[Markovitch and Scott 93] propose an information filtering framework for evaluating and specifying selection mechanisms in machine learning systems. As shown in Figure 17,

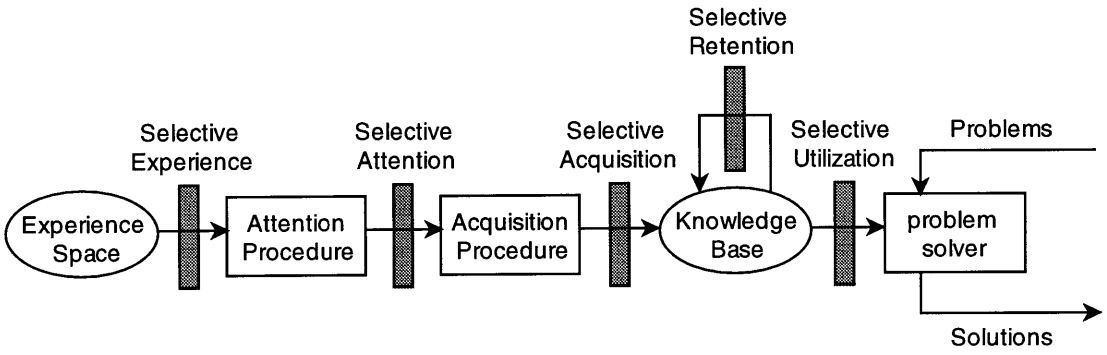

Figure 17: Markovitch and Scott's information-flow filtering model

they propose five different places in which filtering may be employed as a selectional mechanism:

- *Selective experience* reduces the acquisition of knowledge when the number of possible training experiences is large.
- *Selective attention* reduces the acquisition of knowledge when individual training experiences are complex.
- *Selective acquisition* reduces knowledge that has been acquired from some training example(s) from reaching the permanent knowledge base, but can be somewhat limited by not knowing how the knowledge might be used.

- *Selective retention* allows "forgetting" knowledge that will not be worth its storage cost (or is actively harmful), for any later problem.
- *Selective utilization* allows "ignoring" parts of the knowledge base that will not be useful for solving the current problem.

Using this framework, they summarized several well-known machine learning systems. Figure 18, on page 106, is from [Markovitch and Scott 93], amended to include the current research. According to their framework, the work described in previous chapters employs *selective attention* (in the form of how sensory information is accepted, e.g., sensory pruning) and *selective utilization* (in the form of which schemas are updated or spun off from, e.g., cognitive pruning). Selective utilization, as is used here, is not often used in current machine learning systems; IB4 [Aha and Kibler 91] and EGGS [Mooney 89], both relatively recent systems, make use of it, but few others.[1]

[Kaelbling and Chapman 90] propose a technique (the G algorithm) for using statistical measures to recursively subdivide the world known by an agent into finer and finer pieces, as needed, making particular types of otherwise intractable unsupervised learning algorithms more tractable. One could view that as an example of perceptual selectivity: the agent gradually increases the set of state variables that are considered, as needed, when selecting actions and learning (updating statistics).

Along these lines, [Moore and Atkeson 93] propose a technique called *prioritized sweeping*. This approach concentrates learning effort in those regions of the world that are likely to be least well-understood, creating a tree of which questions should be answered and in what order, and shows promise in substantially decreasing the computational com-

1. In IB4, which functions similarly to memory-based reasoning [Stanfill and Waltz 86], instances which perform poorly are simply discarded from the database (selective retention), and newly-acquired instances are prevented from contributing to decisions until sufficient evidence has accumulated to demonstrate that they are reliable (selective utilization). In EGGS, an explanation-based learner, the system only uses learned rules that completely solve a problem, and no use is made of learned knowledge to prove subgoals.

System	Filter	Description	Evaluation Metric
Checker Player [Samuel 59]	Retention	Discards least useful board position	Frequency of use
Genetic algorithms [Holland 86]	Retention	Randomly retains elements with probability proportional to their fitness	Fitness defined in a domain-specific manner
MetaDENDRAL [Buchanan and Feigenbaum 82]	Acquisition	Attempts to find smallest set of rules that accounts for data	Rules that correctly predict peaks not predicted by other rules score higher
Version Space [Mitchell 82]	Experience	Chooses experience that will reduce version space by greatest amount	Selects experience that comes closest to matching half remaining hypotheses
ID3 [Quinlan 86]	Experience	Selects only misclassified instances	Correctness of classification
INDUCE [Dietterich and Michalski 81]	Acquisition	Eliminates candidate generalizations using several evaluation criteria	Includes coverage, specificity, and user defined function
LEX [Mitchell, Utgoff, and Banerji 83]	Experience	The Problem Generator constructs new practice problems	Prefer problems that will refine partially learned heuristics
	Attention	The Critic marks positive and negative instances in the search area	Select search steps on the lowest cost solutions as positive
MetaLEX [Keller 87]	Retention	Removes subexpressions that are estimated to be harmful	A weighted combination of estimated cost and estimated benefit
DIDO [Scott and Markovitch 89]	Experience	Performs experiments on classes with high uncertainty	Prefer experiences involving objects of classes with higher uncertainties
PRODIGY [Minton 88]	Experience	Generates experiments when discovers incomplete domain knowledge	Incompleteness
	Attention	The OBSERVER selects training examples out of the trace tree	Training example selection heuristics eliminate "uninteresting" examples
	Acquisition	Estimates utility of newly acquired control rules and deletes those unlikely to be useful	Eliminate rules whose cost would outweigh saving, even if always applicable
	Retention	Empirical utility validation by keeping the running total of the costs and frequency of application	Estimated accumulated savings minus accumulated match cost; if negative, discard rule

Figure 18: Selection mechanisms in some existing learning systems (Sheet 1 of 2)

System	Filter	Description	Evaluation Metric
MACLEARN [Iba 89]	Attention	The macro proposer uses peak-to-peak heuristics	Propose only macros that are between two peaks of the heuristic function
	Acquisition	Static filtering; only macros estimated to be useful are acquired	Redundancy test (primitive) and limit on length and domain specific test
	Retention	Dynamic filtering; invoked manually	Frequency of use in solution
FUNES [Markovitch and Scott 88]	Retention	Various heuristics to decide what macros to delete	Random, Frequency of use x Length
CLASSIT-2 [Gennari 89]	Attention	Attributes with low salience are ignored	Salience
Hypothesis Filtering [Etzioni 88]	Retention	Runs a test on sample population; passes only hypotheses which are PAC	For a given ε and δ, compute an upper bound on the distance between the hypothesis and the target concept
IB4 [Aha and Kibler 91]	Acquistion	Acquires misclassified instances	Correctness of classification
	Retention	Removes instances that appear to be noisy	Confidence interval of proportions test
	Utilization	Only instances that have proved reliable are used for classification	Confidence interval of proportions test
EGGS [Mooney 89]	Utilization	Learned macros are used only if they solve the problem	Macros that do not solve the problem are worth nothing
This research [Foner 94]	Attention	Sensory bits not relevant to typical world behavior or current goals are not perceived	Spatial and temporal proximity (goal-independent); relevance to current goal (goal-dependent)
	Utilization	Schemas which do not make predictions concerning currently useful sensory bits, actions, or goals, are not updated or spun off	Spatial and temporal proximity (goal-independent); relevance to current goal (goal-dependent)

Figure 18: Selection mechanisms in some existing learning systems (Sheet 2 of 2)

plexity of many common learning situations. They demonstrate it in system that learns a world model in a world of stochastic Markov chains; it is somewhat similar to DYNA [Sutton 90] .

As another way to look at the problem, consider classifier systems, such as in [Holland 86]. Classifier systems have a built-in mechanism for generalizing over situations as well as actions and thereby perform some form of selective attention. In particular, a classifier may include multiple "don't care" symbols which will match several specific sensor data vectors and actions. This makes it possible for classifier systems to sample parts of the state space at different levels of abstraction and as such to find the most abstract representation (or the set of items which are relevant) of a classifier that is useful for a particular problem the agent has. [Wilson 85] argues that the classifier system does indeed tend to evolve more general classifiers which "neglect" whatever inputs are irrelevant.

Others have also addressed the problem of finding the proper tradeoff between *efficiency* (the number of measurements a robot must take, for example) and *accuracy* (number of prediction errors) when attempting to build a world model. For example, [Tan 93] proposes a unified framework for learning from examples, based on four distinct spaces in concept learning: example space, feature space, concept space, and concept description space. This framework is applied to analyzing CS-ID3 and CS-IBL in detail, which are *learning-cost-sensitive* (hence "CS-") algorithms which can trade off accuracy for efficiency in decision-tree-based (CS-ID3) or instance-based (CS-IBL) learning.

5.1.3 Selective attention as goal-driven learning

There has been considerable work in goal-driven learning in recent years. For example, Ram, Leake, Cox, and Hunter have between them produced on the order of thirty papers quite recently which all bear in some way or another on this topic. Some of them are dis-

cussed below; other relevant papers include [Ram 90a], [Ram and Cox 90], [Cox and Ram 94], [Ram 90b], and [Cox and Ram 92].

For example, [Leake and Ram 93] describe aspects of goal-directed learning from the perspectives of AI, psychology, and education in a survey paper that reported on a workshop involving participants from all three areas. They summarized four taxonomies of learning goals: by overarching tasks, by knowledge gap or failure-necessitated learning, by the learning results, and by the learning activity. They also talk briefly about how goals for learning arise, how they affect the learning process, how different types of learning goals relate to each other, and how they are represented.

[Ram and Leake 94] propose a general framework for describing goal-driven learning systems. This survey paper discusses how goals guide task performance, task learning, and knowledge storage, with a strong emphasis on using plans to manage the growth of complexity in all these areas. They propose a two-step framework for managing the learning, in which the first step attempts to reach some particular goal, maintaining a trace of the reasoning performed. Plan failures or deficiencies during this reasoning are then used in the second step, which uses credit/blame assignment to find the source of the failure. Thus, learning in the second step is guided by a knowledge of *what* must be learned and *why*, stemming from the information resulting from analysis of plan failures. Such explicit reasoning about goals is also discussed in [Ram and Hunter 92].

In addition, they point out the importance of *multistrategy learning*. Large, complicated learning systems that operate on real-world problems are increasingly being implemented as multistrategy learners. Such a technique allows using the appropriate learning strategy for the particular piece of the problem which is currently of interest, but are often hard to control or program without some automated way of determining when to use particular algorithms. Learning systems that can reason about their goals and use this informa-

tion to select particular learning modules can make multistrategy learning more feasible. In addition, they increase the feasibility of systems that can actively seek out additional sources of information, rather than having to be spoon-fed information from a small number of hand-picked sources. For example, reasoning involving multistrategy learning is employed by [Ram, Narayanan, and Cox 93] in a system that learns to troubleshoot and is based on observations and a model of human operators engaged in a real-world troubleshooting task.

In recent work, [Hunter 94] provides two examples of these strategies in action. Both are drawn from the domain of biology, which is becoming increasingly important as a source of real-world applications of machine learning due to the large number of interesting datasets, the possibility of external verification and grounding of results via physical experiments, and the discipline imposed by having to cope with very large and scaled-up systems from the start. Molecular biology is also increasingly in need of advanced computational tools to accomplish knowledge discovery.

The first of these examples concerns situations in which there is too *little* data for many learning systems to operate effectively. Hunter's example in this case concerns determining the causes of lethality in osteogenesis imperfecta, a sometimes-fatal bone disorder involving point mutations in the amino-acid coding sequences for collagen. The dimensionality of the space is vast (approximately 243-dimensional), yet only approximately 70 relevant sequences have been determined from sufferers of the disease. With such a small dataset, conventional clustering techniques are useless. However, systems such as RELIEF [Kira and Rendall 92], a Focus/Induce/Extract system, can attempt to eliminate irrelevant features in the dataset using statistical tests. (The system described uses C4.5 [Quinlan 86] to extract condition/action rules from the resulting decision trees; see also [Dietterich 89] and [desJardins 92] for related work.)

The system described by Hunter has successfully discovered previously-unknown information about this disease. It has a large collection of machine-learning algorithms in it, and determines which of them to use for the part of the problem at hand by consulting lists of *preconditions* and *applicability* conditions: a tool's preconditions must be completely satisfied for it to be eligible, whereas a tool's applicability conditions are used to help determine which of the eligible tools to use (in general, the cheapest, fastest such tool is chosen).

The second example concerns situations for which there is too *much* data, such as in megaclustering of protein sequences. The current database set being produced by biologists consists of approximately 100,000 sequences, comprised of 20 million amino acids, and doubles about every 18 months. Hunter's system actively determines, based on the current subgoals, which of multiple data sources to contact over the network, how to communicate with the variety of different databases, what sort of analytical tools should be used to analyze the data, which platform(s) to use to do so (since some tools require very large machines, whereas others do not), and so forth.

This system, too, has discovered new scientific results,[2] which is a strong claim of the utility of the reasoning techniques employed.[3]

In both of these examples, and in the goal-directed learning literature in general, the problem of learning has been transformed from a search problem to a planning problem. A naive approach is quite likely to result in simply turning one intractable problem into another, so this transformation should not devolve into first-principle planning, but should instead yield what Hunter calls a *discovery strategy*, or skeletal plans for learning, an espe-

2. Resulting in publications in the biological literature of its discoveries.
3. Hunter also makes the point that such a multistrategy approach is similar in spirit to the multiple-competence-modules model proposed in the Society of Mind [Minsky 86]; this point is taken up again in the next section.

cially important feature in learning systems that must query databases—since otherwise the expected outcome of any plan is sufficiently unpredictable that almost all plans of non-trivial length will fail.

Such transformations show great promise in making machine learning in large, real-world problems both tractable and useful in true knowledge discovery applications.

5.2 Related Work in Cognitive Science

5.2.1 Introduction

The cognitive science literature about attentional processes is vast. This overview will examine some of the high points of that literature that seem most salient and that seem to be of most utility in producing ideas about implementations of focus of attention in a machine learning system.

One of the most interesting things about the literature of the last two or three decades is that many of the same questions have been asked for all that time. It is slowly becoming apparent that, in some cases, the questions themselves are likely to be misguided; in others, while it has become clear that certain mechanisms are *not* responsible for attention, it is still unclear which mechanisms *are*.

The confusion exists on many levels, from what constitutes a reasonable theory (e.g., there is disagreement about whether it need be computational, in the sense of [Marr 82]) even to simple aspects of terminology (many have railed against the vague use of supposedly well-defined terms, and, amusingly enough, [White 64] dedicates an entire and rather delightfully readable book to philosophical definitions of terms such as "attention," "realizing," "noticing," and so forth).

Many people have summarized important parts of this literature and the questions surrounding it, with an eye towards computer modeling of the mechanism (one example would be [Chapman 90]) or to debate and clarify certain of the confusions of the field (see, for example, Allport's excellent retrospective [Allport 90]). The debates over whether the questions even make sense are not particularly new; for example, many of the concepts and ideas that [Kinchia 80] feels necessary to bury still have enough life in them that [Allport 90] is still driving stakes through their hearts. [Van der Heijden 92] spends an entire chapter of a book doing likewise, exploring and denigrating theories such as the belief in [Broadbent 71] (for example) of *central* and *limited* capacity.

Such guides to the other literature have been very useful in determining how the field has progressed, and in which directions to proceed in examining this enormous array of work; in some of the discussion that follows, I am particularly indebted to the keen and sometimes provocative thinking of Allport and Kinchia.

A large part of the problem with much of the literature on attention is due to its treatment of "attention" as a single, unitary, central process, rather than as a variety of cognitive mechanisms that mediate human information processing. [Kinchia 80] proposes several illustrative theoretical processes, summarized as:

- *All-or-none attention model*, and *weighted integration model.* These models posit a sort of zero-sum information processing paradigm, in which any attention devoted to one stimulus necessarily robs attention from an unattended stimulus. The former model assumes that this works like a switch—attending to one stimulus essentially completely ignores another stimulus—whereas the latter assumes a sort of linear transfer function between two stimuli, where attention can be "shared" between them, albeit with lower processing efficiency for each.

- *Serial coding models.* Many influentual models of perception characterize the initial internal representation of a stimulus as being held in a sensory register, which is assumed fairly rich but unprocessed, and which must be processed relatively quickly lest it decay while in temporary storage in this register. Some view attention in this regard as a switch [Broadbent 58] or a filter [Treisman 69] which determine what information was retrieved from the register for further processing. [Rumelhart 70] characterizes this mechanism as a feature-extraction process, in which individual features are serially extracted and coded from this register.

Note that much of the debate about a single locus of attention seems rooted in more fundamental conceptions of how the mind works, in areas unrelated to selective attention per se. There is a long history of dealing with the mind as if it possessed a homunculus somewhere, leading to theories whose explanatory powers are negligible. Dennett and Kinsbourne, for example, feel this problem keenly [Dennett and Kinsbourne 92]. In reporting research results concerning the perception of subjective time, they spend considerable effort first demolishing the *Cartesian Theater* model of the mind, in which the mind is presumed to have some place where "it all comes together." Their research instead supports what they call the *Multiple Drafts* model, in which discriminations in multiple modalities are not registered and synchronized before "presentation" to "consciousness," but instead are distributed in both space and time in the brain. The arguments that they present in demolishing the Cartesian Theater model are of the same sort required to demolish the "single, serial" model of attentional processes, as delineated below.

Two of the major aspects of the problem concern arguments over *early* versus *late* attention (e.g., whether attentional selection occurs before or after stimuli are coded into categories), and which cognitive processes require attention, and are hence limited by attention, and which do not. The major thrust of these arguments is to determine possible

constraints in the processing architecture of the brain, so as to determine the overall and detailed architecture of how the brain processes information. Unfortunately, many of these architectural models are imperfect at best, and are surprisingly unhelpful in generating useful architectures with sufficient explanatory power to permit either further analysis of the system, or its reproduction (e.g., as a program).

The vast majority of studies of attention concentrate on visual attention. Of those, many are detailed neuroanatomical studies of either humans or other primates (often via lesion studies or examination of pathological cases). Others are performance tests of healthy human volunteers. A small percentage deal with auditory attention, with an almost insignificant percentage examining other forms of attention. This means that examples of attention are heavily biased towards human (or at least primate) visual attention only.

Such studies of attention are examining a system (namely, human cognition) which is far more complicated than those yet investigated in machine learning. Consequently, while they serve as interesting inspirations for approaches to try, it is not claimed that the research in this thesis either explains anything about mammalian visual attention, or that such studies necessarily will lead to a direct implementation.

5.2.2 The plausibility of attention as a system of limitations

There is a very widespread view that the need for selective attention stems from fundamental limitations in cognitive processing power in particular portions of the brain, and that, if the brain were to have infinite computational power, such attentional limitations would be unnecessary. This is argued over a span of decades by [Broadbent 58] [Broadbent 71] [Broadbent 82], among many others. He and others view attention itself, therefore, as a limited-capacity system, one which must be shared by many processing stages and whose capabilities are therefore competed for by various portions of the brain.

This view also espouses that some tasks are "automatic" and hence do not require attention, presumbly by using portions of the brain whose capacity is not as severely limited [Kahneman 73]. The appeal of dividing cognition up in this rather intuitive fashion, of course, is that, if one could identify the bottlenecks, one might begin to get a handle on how cognition is structured.

Allport [Allport 90] describes Treisman's feature-integration theory (FIT) [Treisman and Gelade 80], [Treisman 88] as one of the best-known of the theories that equate attentional mechanisms with intrinsic bottlenecks in processing. It includes careful characterization and theoretical arguments that, according to the theory, necessitate serial focusing on each item to be perceived in turn in order to correctly perceive objects that must be distinguished by conjunctions of separable features. Yet, immediately after describing Treisman's (and others') theories of attention, Allport (quite rightly) takes issue with much of the terminology of the field; even what is meant by the word *selection* is ambiguous: Does it mean "any task-dependent modulation of sensory neuronal responses?" "Selective facilitation?" "Attentional tagging?" "Selective feature integration?" "Entry to a limited-capacity short-term memory store?"

The term *attention* has similar problems in cognitive science: [Johnston and Dark 86] ask whether attention is some hypothetical causal agency which can be directed or focused on an entity (with the result that this entity may be "selected"), or an *outcome*, characterizing the behavior of the whole organism. They point out that most current attentional theories postulate the above hypothetical causal agency, but that there is a great deal of drift between the two concepts; they also mention that, in most contemporary theories, this causal agency "has all the characteristics of a processing homunculus," which does not help us to understand the underlying mechanisms.

Many of the assumptions about cognitive architecture adopted by models of attention appear in the following list, adapted from [Allport 90]:

- Information processing follows a linearly ordered, unidirectional sequence of processing stages from sensory input to overt response. Parallel or reciprocal processing is disallowed in this model.
- Such a sequence is already known, or can be assumed a priori.
- The processing of nonsemantic attributes occurs before processing of semantic attributes.
- Spatial attribute and relation processing logically precedes categorical or semantic distinctions. There is just *one* locus of attentional selection, hence it can be *early* or *late*, but not both.
- Attentional selection therefore serves as a gate for *any* further processing to be performed; whatever does not make it past this gate will be remain unprocessed.
- There exists a single "central system" of limited capacity, responsible for all cognitive processes that "require attention," which can only be bypassed for "automatic" processes (defined, of course, as those which do *not* "require attention").

I will detail below only a few of the ways in which, as mentioned in the introduction to this section, this set of assumptions has begun to fall apart. But as an overall trend, there is growing pressure to develop a theory of attentional selectivity and *control*, rather than the current conception of attention as being a passive information *filter*.

Let us start at the top. If processing is inherently serial, why does the brain seem to have separate processing for "what" versus "where" information? Consider the primate visual system, composed of at least twenty different modules [Desimone and Ungerleider 89] [Ullman 91]. These modules can be broadly grouped into the *ventral* system, crucial for

form-based object recognition, and the *dorsal* system, responsible for spatial vision and coordination [DeYoe and Van Essen 88] [Ungerleider and Mishkin 82]. This "what" versus "where" system is quite well established in the literature of visual attention, and poses a rather embarrassing problem for the "single, serial" assumption above. (Indeed, [Felleman and Van Essen 91], to pick only one of many similar papers, demonstrate that among 32 areas that are associated with visual processing in the primate visual cortex, approximately 40% of all possible connection pathways between the modules actually exist! This makes cortical visual processing organization look more like a bush than a hierarchical or serial system, and does not even include the straightforward reciprocal neural connections in each cortical area; see the discussion immediately below.)

Worse yet, almost everywhere in the cortex, the "forward," *afferent* connections that one would expect (leading from the retina toward higher centers of processing) are paralleled by equally rich, "backward," or *efferent* connections [Ullman 91]. If all processing proceeds in the afferent direction, what are all of those reciprocal connections *doing* there? Many have proposed ideas: for example, [Mumford 91] proposes that each cortical area is responsible for updating and maintaining knowledge of a specific aspect of the world, at any given level from low level raw data to high level abstract representations, and that the multiple, often conflicting hypotheses which result are integrated by thalamic neurons and then sent back into the cortex, making the thalamo-cortical loop a sort of "active blackboard" system and thereby explaining the density of reciprocal cortical connections.

Ullman [Ullman 91] has proposed a particularly interesting idea with his sequence-seeking counterstreams model, in which he posits that the neocortex searches for mappings between "source" and "target" representations, exploring both "top-down" and "bottom-up" a large number of alternative sequences in parallel.[4] Finally, even though most diagrams of the visual cortex show each module interacting with a few nearby ones in a semi-

well-behaved processing mesh, almost every module *also* has a direct connection (e.g., an output pathway) to some motor or action system, forming a number of direct, parallel links between sensory and motor systems that could potentially bypass all levels of higher processing [Creutzfeldt 85]. What are those links doing there, if the "single, serial" model is correct?

It has also been argued by many that tasks conforming to what [Kahneman and Treisman 84] call the "filtering paradigm," in which the to-be-selected visual items are cued by *non*spatial visual attributes such as color or size, may instead depend on selective cueing by *location* [Butler and Currie 86, Johnston and Pashler 90], and that where this spatial separation is absent, performance drops [Johnston and Dark 86]. This tends to imply that many "early selection" paradigms of visual attention may instead correspond to spatial selection.

But the picture is murky even in spatial selection. For one thing, extensive experimental evidence reveals many *different* coordinate systems and corresponding transformations along the path from the retinotopic input through the cortex. For example, [Allport 90] provides a virtual laundry list of such transformations, mentioning some that take account of eye and head position, some that code location in terms of arm- or body-centered coordinates, and some based on environment- and perhaps object-centered coordinate systems; a small sampling of work in this area can be found in [Andersen 87] [Andersen 89] [Ellis et al 89] [Feldman 85] [Hinton and Parsons 88] [Marr 82] [Soechting, Tillery, and

4. Such counterstream architectures, if they exist in the brain at all, may no longer be unique to it, however. Bob Sproull of Sun Microsystems has proposed [Sproull 94] a novel, "counterflow pipeline" architecture for advanced, pipelined RISC CPU's which shares many remarkable features with Ullman's counterstreams model of processing. Instructions and results propagate in opposite directions in a processing ladder, interacting with each other as they pass, and employ only local interaction (e.g., only within a ladder level, or between two adjacent rungs of the ladder). Such a design also admits an asynchronously-clocked implementation, making it more similar to possible cortical models such as Ullman's. However, the intersection between cognitive science and machine architecture is understandably not what it could be: neither Ullman nor Sproull had heard of the other's work.

Flanders 90] [Zipser and Andersen 88]. This does not even include the many lesion studies which investigate neglect in various coordinate systems after brain damage.

If recent results seem to put nails in the coffin of attention as a serial, feedforward, strictly limitation-based strategy, what models are proposed instead? [Allport 90] argues that the influence of attention in noncategorical, spatial-vision systems is of the form of *enhancement* of neuronal responsiveness in attended locations, rather than attentuation of unattended locations, and that many results involving delays in attending to stimuli reflect the time cost of disengagement from the cued location, rather than withdrawal of processing resources from the uncued location. (He also notes work, such as [Driver and Tipper 89], which points out the problems with equating "no processing" and "no interference.")

Indeed, lesion studies such as in dorsal simultanagnosia, in which the patient perceives only one part of any given object even though his visual field is often full and complete, seem to indicate that such damage leads to an inability to *disengage* from one part of the visual field in order to shift attention to a different part of it: unilateral lesions [Posner et al 87] [Morrow and Ratcliff 88] can lead to problems shifting attention to the contralateral side, and full simultanagnosia can lead to problems shifting attention in *any* direction [Luria et al 63].

Viewing attentional processes as a process involving commitment of resources, rather than filtering, leads [Crick and Koch 90], for example, to suggest that attention facilitates local competition among neurons: in other words, when a local group of neurons is not attended to, it can have multiple (ambiguous) outputs, but attention then narrows down the possible outputs, forcing disambiguation. This view of attention is quite different from that of protecting the limited computational power of a single center from overload.

This argument for a multiplicity of attentional mechanisms, each "specialists" in a particular cognitive area, fits in nicely with the Society of Mind hypothesis [Minsky 86]. While Minsky posits that many of the aspects of attention have to do with limits (e.g., limitations in processing leading to the intuitively serial feeling of thought, or limitations in a particular agent's ability leading to inability in tracking multiple locations simultaneously), the theory does not say that there is a *single* limit anywhere—only that different agents will likely contribute different limitations.

The existence of multiple loci of attentional control are reinforced dramatically by [Mangun, Hillyard, and Luck 90], who use a combination of MRI brain images, behavioral data, and event-related brain potential mapping. While this research comes down rather strongly on the side of "early" selection (since the effects cited occur very quickly, within 150 msec), [Sperling, Wurst, and Lu 90] introduce a new theoretical construct, attentional "tags," through which visual item traces may be selected from short-term memory, rather than positing a single filter. Such an interpretation is completely in support of multiple loci of attentional control.

It is interesting to note that the majority of even current work in the machine learning community still treats attention as a single, serial pathway, and structures its systems accordingly. (See, for example, Figure 17, on page 104, from [Markovitch and Scott 93], and the discussion in Section 5.1, on page 102.)

One reason for this might be that current machine learning systems are still too primitive to take advantage of architectures rich in reciprocal connections, or that contain multiple loci of control or information processing. For example, the bulk of this thesis concerns itself with single-agency pruning, of the type of "limitations and bottleneck" school so denigrated above. In addition, the sensory and cognitive system modelled in this research is greatly simplified compared to even the most rudimentary levels of human cognition;

many insects might have more sensory processing abilities, and even the simplest mammal must be better at memory and generalization.[5]

Just because the evidence for a single, serial control of attention no longer appears to be compelling is no reason, of course, to discount much of the work that has been done in attention. While it may not be the case that an explanation of one particular aspect of attention explains all of attention (either in that modality or others), there are many probably-correct explanations of parts of the attention puzzle. Unfortunately, few have implemented their theories, possibly because many of them are insufficiently precisely described for such implementation. This makes it even more difficult to determine which theories might be correct.

For example, [Chapman 90] cites several aspects of visual attention, such as results in covert attention [Posner et al 80], and the winner-take-all addressing pyramid in [Koch and Ullman 85] in support of visual spotlight search behavior. [Treisman and Gormican 88] have done extensive research on visual pop-out behavior in visual search routines; for surveys of visual search in general, see [Julesz 84] and [Treisman and Gelade 80]. But Koch and Ullman did not implement their theory; in fact, Chapman's work is one of the few to implement several subtheories of visual attention.

It may be, as machine learning systems become more sophisticated, employing multiple processing strategies in a richly-connected information architecture, that they will be better positioned to take advantage of current thinking about attention in cognitive science.

5. Although the lack of generalization in the schema system, as currently designed, does seem to put it on a par with certain insects. For example, bees apparently remember places retinotopically—if they learn a shape with one part of their eye, they can only recognize it again with that same part [Christensen 94]. Bees, which have magnetite in their abdomens as part of their navigation system, face magnetic south (or magnetic northwest in certain cases in which south is infeasible) when encountering and departing targets of interest. By doing so, they can image the target in the same orientation; rather than rotating a mental representation of the target, they simply rotate their real eyes instead until a match is acquired. Artificially imposed external magnetic fields lead to predictable perturbances of this behavior.

Organizations such as the subsumption architecture [Brooks 86], for example, or the Society of Mind [Minsky 80] seem as if they will be logical computational testbeds for implementing computational verification of multiple-loci attentional models.

Indeed, as shown by the some of the systems mentioned in Section 5.1.3, particularly the multistrategy systems of Hunter, Ram, Cox, and others, the increasing complexity of modern learning systems is forcing implementations of the control of their attentional focus down just the sort of pathways that the multi-locus models of modern cognitive science might lead one to expect.

Chapter 6: Conclusions

6.1 The effectiveness of focus of attention

The results described earlier in this thesis demonstrate that mechanisms for focusing attention can greatly decrease the amount of work required by a learning system. Both goal-independent and goal-dependent focus methods can be employed simultaneously, decreasing the work of learning (and hence increasing its speed) by taking advantage both of invariant characteristics of the world and of guidance provided by the set of goals that the agent must achieve.

Because these methods can decrease the rate of growth of a fundamentally $O(n^2)$ process, their effects only increase as the size of the problem or the number of known facts increases. This research showed combined improvements of over a factor of 50; longer runs would have shown even more.

In addition, it was demonstrated both qualitatively and quantitatively that the correctness and completeness of the learning performed in the systems studied was not impaired by these techniques. One pays a price for them, namely having to perform more experiments in the focused case to learn roughly comparable amounts of knowledge about the world, but this price is quite small compared to the increase in efficiency that results. It is possible that there are many other systems which can utilize similar techniques to achieve faster learning without substantially sacrificing correctness or relative completeness.

As mentioned in Section 3.3.1, on page 49, and in Section 4.4.1, on page 86, the results presented here are primarily from the infant/eyehand scenario. However, a smaller number

of similar runs have been performed in the Hamsterdam scenario, with comparable results. Given the approximate similarity of sensor systems and action repertoires in the two scenarios, this is unsurprising, but reassuring. The Hamsterdam scenario additionally offers the potential for more interesting goal sets, due to the more dynamic world available—there are many opportunities available there for more sophisticated experiments involving goals and their control.

In short, using focus of attention is one of the many techniques that can and should be employed to allow autonomous agents to learn more about their environment with less computation. This can allow certain applications, which formerly ran too slowly to be practical, to be run at more reasonable speeds.

6.2 Future work

There are many ways in which this work might be improved or extended. A representative sampling of such ideas follows.

This is clearly far from an exhaustive list. Indeed, viewed in a larger context, the questions from [Maes 94] are still very much with us. Focus of attention cannot hope to address all of those questions, but many of them might be partially answerable by using more sophisticated focus mechanisms. This is an area deserving of future investigation.

6.2.1 Generalization and abstraction

One of the most frustrating aspects of the current schema system concerns its inability to generalize in certain ways. The synthetic item machinery allows one form of generalization, which is important for shifting to different levels of abstraction, but the implementation used in this research lacks composite actions, which severely limits the sort of generalizations that might be made. Even with this machinery in place, simple generaliza-

tion across a category of input would be very useful;[1] currently, the learning system requires several examples at *each* point in the state space and therefore does not perform this sort of generalization. This research has not addressed the details of what would be required to function effectively in a learning system which employed such types of generalization.

The focus system as currently implemented does not explicitly address different levels of abstraction in the learning system. A system which created explicit categories at various levels of abstraction would require some form of support for focus; how to do this well is an open problem.

6.2.2 Filtering

In addition, the overall focus mechanism as implemented decides how to *filter* its perceptions, cognition, and actions based on characteristics of the domain and its current set of goals, analogously to certain ideas about attention as a system of limitations that were questioned in Chapter 5. It does not reason at a metalevel about the goals themselves,[2] nor does it use multiple concurrent learning mechanisms or multiple simultaneous processing pathways. As such, there are large opportunities for further work using multistrategy learning and recent, non-filter-based, multi-locus ideas from cognitive science. Such work could enable a more sophisticated action selection system.

1. An example would be automatically inferring that, if moving the eye right causes an object to appear to slide left from one particular visual location to another (e.g., by turning off one visual item and turning on the visual item to its left), then this would be true at *all* points in the retina. Such generalization would require a retinotopic map (e.g., not the unordered "bag of bits" currently employed) so that concepts such as "to the left of" could be inferred without exhaustively acquiring data about every adjacent pair of visual items. Without such a map; there is no way of even determining adjacency without such exhaustive experimentation. Such a retinotopic map *is* assumed in particular goals defined here (e.g., we assume that we know a priori, due to hardwiring, which coarse visual items actually correspond to the fovea, in certain goals), but this is not a general mechanism and cannot really be used by the schema system per se in order to increase its representational power.
2. For example, to change their hardwired mapping from the goal to the allowed set of percepts and actions.

6.2.3 Experimental strategy

In addition to the above ideas about the action selection system, a more intelligent experimentation strategy would be welcome. In particular, within the currently-allowed set of actions specified by the active goals, actions are chosen randomly. This leads to exploration of the state space in a manner which is probably quite inefficient. Consider the figure below, from [Thrun 94]. In this figure, we have a robot which must simply navigate from

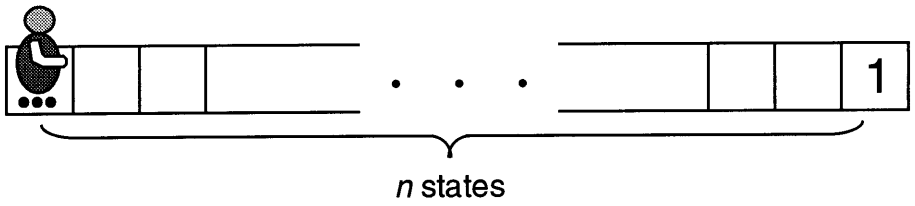

Figure 19: A task which is *O(1)* with counters, and $O(2^n)$ without

one end to the other, *without* learning. If the robot takes steps randomly either to the left or to the right, its expected time to reach the goal is exponentially bad, e.g., $O(2^n)$. If, on the other hand, the robot is allowed a *counter-based* approach in which it simply drops a counter on each visited square, and picks a square without such a counter when it can, its performance improves to *O(1)*. Indeed, it has been proven that, subject to some simple and common assumptions[3] any learning technique based on random walk is inefficient in time [Whitehead 91a] [Whitehead 91b]. On the other hand, even a very simple strategy such as "go to the least visited neighboring state" can reduce this inefficiency from exponential, e.g,. $O(2^n)$ time to polynomial, e.g., $O(n^2)$ time, regardless of whether or not one has a model that can predict the next state from the current one [Thrun 92] [Thrun 94].[4]

3. These are: a state space which is finite, deterministic, and ergodic (e.g., no states from which, once entered, the agent cannot escape), in which the agent receives a reward only in the goal state; there is no information available about the domain a priori; random actions change the distance to the goal state by only +1, 0, or -1, and can be expected to increase the distance to the goal on the average; and, finally, the size of the state space is polynomial in the largest possible distance to the goal state, e.g., the *depth* of the state space (this holds for most state spaces studied in literature, e.g., grids of arbitrary dimensionality).

The system explored in this research does not quite meet Whitehead's conditions. In particular, because of the stochastic characteristics of the domain microworlds studied, in which other entities may move and hence change parts of the state space, and because of the effects of sensor aliasing [Whitehead and Ballard 90], the system is nondeterministic. Similar results do not exist for nondeterministic domains, and for some *malicious* domains, it can be shown that *any* exploration technique will take exponential time to find a goal state [Thrun 92].

Nonetheless, given domains which are not malicious, and which are not *too* stochastic, it is quite possible that some sort of counter-based approach could increase the rate at which relatively-unexplored parts of state space are encountered, hence decreasing the amount of work per generated schema. One possible (but untried) approach would therefore be to use something akin to prioritized sweeping [Moore and Atkeson 93]. An even simpler approach could be to always pick that action A for which the average reliability of all currently-applicable[5] schemas containing action A is minimized.

6.2.4 Goals

The goal system implemented here is unsophisticated. It is unlikely to scale well to large numbers of goals, in part because of its rather nonhierarchical space of goals, and in part because goals and their relations to each other must currently be hardwired. It also offers little support for multiple concurrent strategic (as opposed to tactical) goals, or for sharing work between goals, nor do goals reason about their performance at a metalevel in order to better guide the learning, as is done in some current multistrategy learners [Hunter 94] [Ram and Leake 94]. A system which learned useful mappings from goals to the correct strategy for focus of attention, rather than having such a strategy hardwired in

4. Such a predictive model does help, but the problem is still $O(n^2)$.
5. E.g., context satisfied, meaning that their context agrees with the currently perceived state of the world.

for each goal, would also be quite useful. It would extend the current work from the realm of very simple animals further up the phylogenetic tree, and may help some of the probable scaling issues in the current design.

6.2.5 Occasional defocusing

The focus system's selectively is a bit sharp; it is essentially an all-or-nothing sort of focus. One that occasionally defocused might lead to more opportunistic exploration of the space without undue cost; integrating this into the system in an intelligent way touches upon many of the explore/exploit problems mentioned in [Maes 94].

Appendix A: System Architecture

Figure 20, on page 132, summarizes the architecture of the two testbeds built for this research. In general, the learning system is connected "at arm's length" to the microworld. The learning system can only send simple commands (one of a small number of actions) to the microworld, and it only receives what sensor information the microworld transmits back. It does not have access to internal microworld state.

The learning system is the same in either scenario. As shown in Figure 20, it contains the schema system proper (which consists of, essentially, a reimplementation of Drescher's schema system [Drescher 91] without certain elements),[1] plus the goal-independent focus mechanism described in Chapter 3 and the goal-dependent mechanism described in Chapter 4. The action selection system picks actions at random in both the unfocused reimplementation of the original algorithm and in the goal-independent work described in Chapter 3, but is informed by the goal system in the further work described in Chapter 4. In addition, the learning system contains a large amount of diagnostic and performance-monitoring code, from which the results (in terms of work per schema, etc) described in this research were derived.

The microworld used in each scenario is, of course, different. In the case of the infant/eyehand scenario, it runs in the same process as the learning system, though its only connections to the learning system are via the aforementioned sets of actions and sensory bits. In this case, the entire system was implemented as a single process in Lisp under Symbolics Genera.[2] The microworld itself has no sophisticated rendering apparatus; instead,

1. Such as the composite-action system or the mechanism for computing delegated or instrumental value.

simple character-based diagrams can be produced of its current state for inspection, evaluation, and debugging.[3]

In the Hamsterdam scenario, however, the microworld is a separate process, implemented as C and C++ code using SGI Inventor, and runs on a Silicon Graphics workstation. Hamsterdam includes sophisticated, real-time, three-dimensional graphic rendering. It communicates with the learning system (in this case, implemented in Harlequin LispWorks and running in a separate process, on either the same machine as Hamsterdam or a different one) via a network connection consisting of a UNIX TCP socket.[4] This proved to be implementationally less than ideal, but of no interest theoretically, because the already-enforced "arm's length" relationship between the learning system and any associated microworld already decreases the possible coupling between the learning system and its microworld to a very loose connection.

In both cases, the Lisp portion of the system (learning system and microworld in the infant/eyehand scenario, or the learning system only[5] in the Hamsterdam scenario) could be snapshotted to disk and restored later using a component called the *snapshotter.* This was an implementation convenience to make it easy to duplicate runs with different param-

2. Since the system is coded in Common Lisp, it could run under any Common Lisp implementation, however. It currently runs under Symbolics Genera and Harlequin LispWorks, and ports to other lisp implementations would be extremely straightforward.
3. A simple, colorful, graphical representation of this world was also constructed to demonstrate certain aspects of the early system, but was essentially equivalent to the character-based diagrams in content.
4. Were LispWorks able to directly incorporate the code and libraries of Hamsterdam, the learning system and Hamsterdam could run in a single process. However, because they were required to run in separate processes due to limitations in currently-released versions of LispWorks, there is no reason why any learning system, running on any other machine, could not be substituted for the current configuration.
5. The state of the Hamsterdam microworld cannot be so preserved in the same fashion, having never been designed for it. Thus, strict reproducibility of runs in the Hamsterdam scenario is not possible, due to the differing environments that would be faced even by "identical" runs of the learning system. This makes the Hamsterdam scenario slightly more difficult for debugging, since events would not always unfold identically even if the same code were to be run.

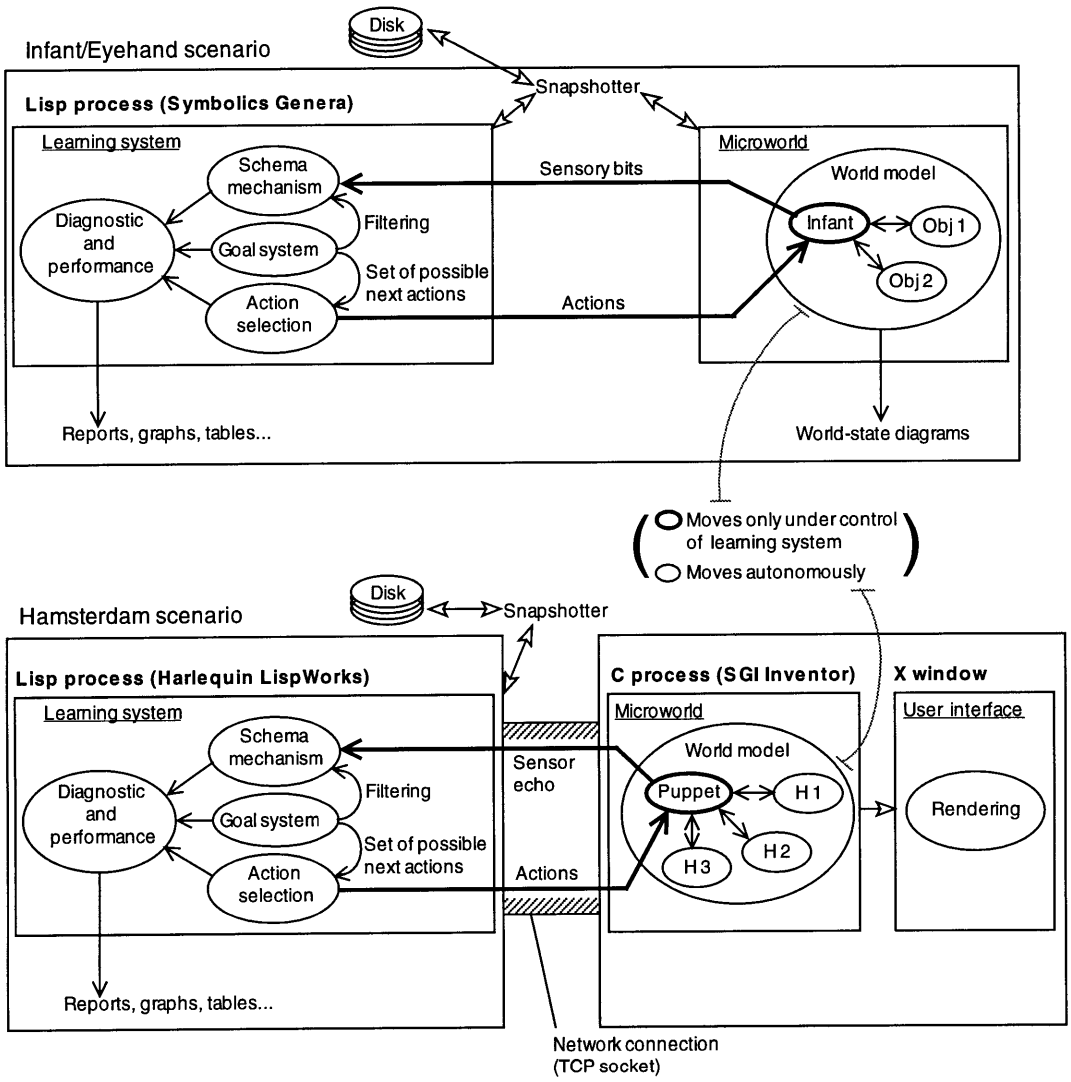

Figure 20: System architecture of both scenarios

In the infant (top) scenario, everything runs in a single process. The only object under control of the learning system is the infant; the other objects in the world occasionally move by themselves.

In the Hamsterdam (bottom) scenario, the learning system runs in one process, and Hamsterdam runs in a separate process. The only object under control of the learning system is the puppet; the hamsters and predators are free to wander around autonomously, and do so continuously.

eters, or to return to a long run after a reboot, given that the unmodified, unfocused system could take up to 3-4 days to produce 3000 to 4000 schemas.[6]

A communications signature of the interactions between the learning system and any given microworld is summarized in Figure 21, on page 134. It is expressed in the typical fashion for network protocols, as a Feynman diagram. In general, there is a lockstep relationship between an action being requested by the learning system, and the corresponding sensory bits being returned by the microworld after the action has been performed. This lockstep relationship, while perfectly reasonable in the infant/eyehand scenario, is less realistic in a more dynamic microworld, such as that employed in the Hamsterdam scenario.

Because of this difference in microworld characteristics, the communications protocol employed for the Hamsterdam scenario was slightly modified. An explicit *null action*, not present in the infant/eyehand scenario, was introduced. Like any other null action, it is able to be part of a generated schema. Strictly speaking, such a null action could simply be modelled as an action that never produces any detectable result, except for taking some amount of time (and, indeed, this is exactly how it was implemented). However, it serves as a placeholder in the schema system to model the *result of not doing anything*, which is itself an important concept in a dynamic world.[7]

6. While the snapshotter's contribution was considerable early on in the course of this research, its utility steadily decreased, because each successive refinement to the learning system, as described in later chapters, served to increase the speed of the resulting system and thereby decrease both the real time and the computational work of producing equivalent states. In some cases, when using the most highly-selective learning, runs of useful size could be produced in an hour or two—a considerable improvement.

7. Consider a robot which must outwit a motion detector. If it never stops and waits, it will never have a chance to observe the little red light on the motion detector go out. This means it will never be able to correlate its motion in the environment with the behavior of the motion detector, since the detector will always be detecting movement, and the light would always be on.

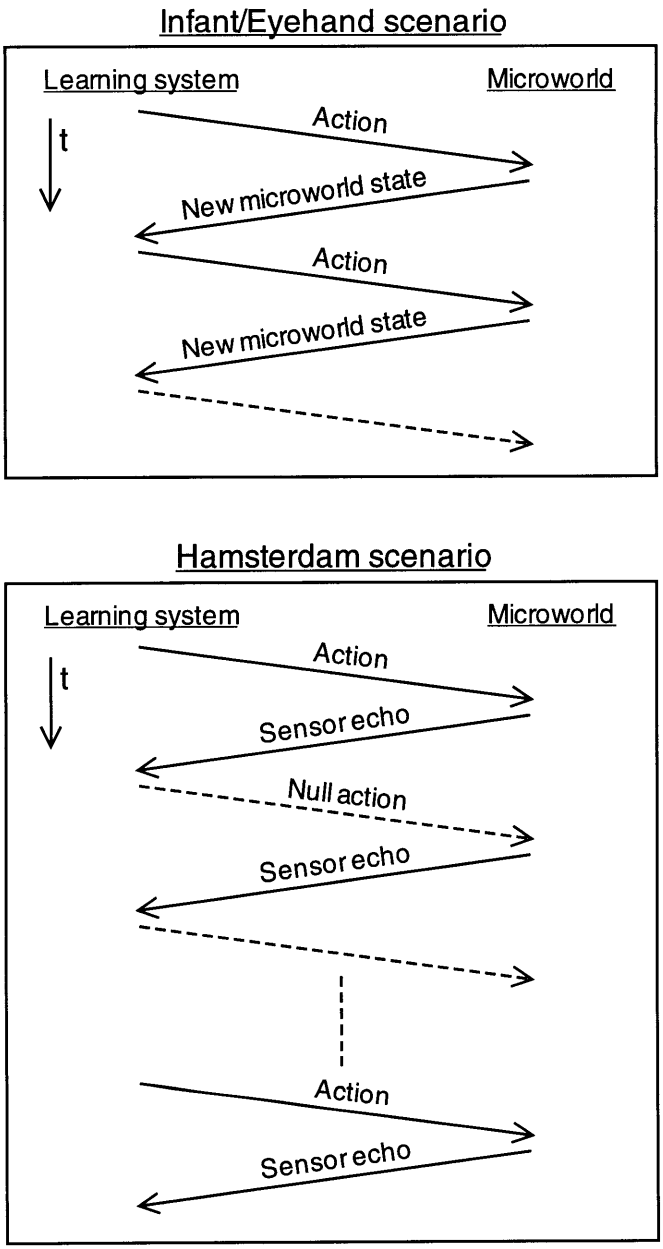

Figure 21: Communication signature for both scenarios

Time flows downward in both diagrams. In both scenarios, the learning system requests an action, has it performed in the microworld, and gets a set of sensory bits returned. In the Hamsterdam case, an explicit null action is also possible.

By explicitly representing null actions, we allow changing the ratio of the number of "real" actions performed to the number of "null" actions performed. Let us define this ratio as follows:

$$\alpha = \frac{A}{\varnothing A}$$

where A is the number of "real" actions taken over some interval, $\varnothing A$ is the number of "null" actions taken over the same interval, and α is therefore a measure of behavior: if α is substantially *less* than unity, the behavior of the system can be said to be *shy* or *inhibited*, whereas if α is substantially *greater* than unity, the system's behavior can be said to be *outgoing* or *audacious,* to use some rather anthropomorphic terms. The parameter α thus behaves vaguely like the level of limbic-system activation in the mammalian brain. When the learning system is outgoing, taking action most of the time and never pausing to watch the world go by, it can learn a great deal about the effects of individual actions it takes, but not how the world functions if it were not taking actions at all. Conversely, when the learning system is shy, not taking action and instead observing how the state of the world changes when it does nothing, it can learn how doing nothing affects the world.[8]

There is a secondary reason why null actions were introduced. The Hamsterdam microworld is continually updating its internal state, because its agents operate in real time, and it must continue to update so as to re-render the scene and preserve the illusion of a dynamic, changing world. Such updating happens several times a second, which is substantially faster than the learning system can keep up with the world on the current hard-

8. If the system were provided with a much more sophisticated source of data about the world, which corresponded to being told about the actions taken by other agents in the world, it would be possible for it to learn the correlations between other agents' actions and their effects. However, no source of data like this exists in the current scenarios studied; to make such a source available, the control structure of the learning system would have to be modified to substitute other agents' actions for its own while computing statistics and producing schemas. Presumably, the right time to make such a substitution would be when the agent is otherwise performing a null action, or some action whose result is "overlearned" and hence whose results are no longer interesting to the learning system.

ware, even with the improvements described in later chapters. Were the learning system to accept every update, it would be forced to discard many of them, or it would lag further and further behind the current state of the world. Yet, the microworld cannot be expected to know how fast the learning system can run. (Any such attempt would be doomed to failure, given that the learning system's running speed varies dramatically based on what sort of focus it employs and how much it already knows, and because both the learning system and Hamsterdam are not tied to particular platforms.)[9]

Null actions provide an explicit solution to this dilemma. By allowing the learning system to dictate exactly when it receives sensory input (it always gets one update after any action, whether real or null, no more and no less), the learning system polls the microworld for sensor updates. Therefore, the microworld's sensor update rate is throttled by the learning system. This is reasonable, since more-frequent updates would be useless to the learning system anyway. If the learning system is quite slow in requesting updates compared to the speed at which the microworld runs, it will miss many important events and may in fact not learn anything useful. Thus, any real agent employing this approach would have to be placed in a situation where its cognitive speeds are up to the task of the world with which it must interact, a familiar problem in both engineering and biology, or would at least require the ability to be infrequently but quickly interrupted if some high-priority sensory signal (such as being about to go over a cliff) demanded prompt attention.

9. The learning system can run under Symbolics Genera, at a variety of speeds depending upon the type of Lisp Machine in use, or under Harlequin LispWorks, again at a variety of speeds depending on the type of SGI used. Hamsterdam can run on many kinds of SGI platforms, each at a different speed.

Appendix B: Implementation of Efficient Planning

B.1 Introduction

This appendix provides some details of how certain parts of the planning algorithms were implemented. Perusal may help to clarify why certain things were done the way they were, and give some insight into where difficulties remain.

These planning algorithms were designed to run essentially nonincrementally, e.g., they do not amortize their effort over the learning lifetime, but do all of it at the end. In a real agent, of course, a system that incrementally built or modified the data structures would be more appropriate.

Note that all of these algorithms were developed for use on a serial machine. Parallel machines may be able to finesse some of these problems by doing their planning in a parallel fashion—at least as long as the problem does not grow larger than the number of available processors, of course.

B.2 Reachability

The first requirement when attempting to plan a path from one schema to another is to be assured that there is, in fact, such a route. Simple explorations of a digraph composed of several thousand schemas will run exponentially slowly and are completely unsuitable; other algorithms, such as the iterative deepening algorithm discussed in Section B.4, on page 140, will fail to terminate if called to compute a path that does not, in fact, exist.

The general idea is to function like a depth-first, mark-sweep garbage collector. We build a *root set* of all schemas with null contexts and non-null results. (Such schemas must

be at the start of any chain, since, with null contexts, they cannot be chained to.) For each root object, we traverse the set of all schemas with non-null contexts. If we find an unmarked schema whose context chains from the result of the root object, we add it to the chain being built, mark the schema we just found, and then use the new schema as the base of a chain to be recursively extended. At any point during this process, we abandon the attempt to extend the chain if we either cannot find a next schema to chain to, or if the next schema is already marked (to prevent loops)—a marked schema must, by definition, have already had its chain traversed, or to be in the process of having its chain traversed at the moment. Since schemas are never duplicated and are therefore guaranteed unique [Drescher 91], we cannot find that the first schema we find to extend the chain is already marked, since the only way that two contextless schemas in the root set have the same next schema in the chain is if their results are identical, and we just assumed that this cannot be. Therefore, every schema in the root set that has any next schema will eventually wind up pointing at such a next schema.

While extending the chain is depth-first, we also go breadth-first in finding other schemas that might extend the chain from this schema for the case where this schema's result might chain to more than one other schema at this level (e.g., if the current schema is A/FOO/B, then valid next schemas are both B/BAR/C and B/BAZ/D, where there are two different actions leading from the same context to [same or different] results [C might or might not equal D]).

When we extend a chain from schema A to schema B, we push B onto a list maintained by A. At the end of this mark-sweep process, each schema contains a list of all other schemas that can possibly be reached, by any path, from that schema. This constitutes the *reachability information*. This is a relatively expensive operation; it can consume several minutes on a fast Lisp processor.[1]

Note that the root set does *not* need to be regenerated when performing or recovering from a lobotomy; this is explained further in Section B.3 below. This means that it may be generated once for some particular set of schemas, and never regenerated unless schemas are added to the knowledge base (schemas can never be subtracted).

The reachability results generated display interesting properties. In particular, there exist "islands" of connections, such that any schema in an island is reachable (via some, possibly quite convoluted and low-reliability) path from any other schema in the island. (This is all the more interesting when one considers that schema chaining generates a digraph, but islands of connections are generally connected in both directions.) As one increases the number of schemas in the knowledge base, the initially large number of small islands decreases, as the average size of each island increases. A knowledge base of 3000-4000 schemas typically displays on the order of half a dozen such islands, which among them completely partition the space.

B.3 Cached (lobotomized) reachable schemas

Given the above reachability information, we can relatively quickly compute a related concept, namely those schemas reachable from some given schema *when the knowledge base has been lobotomized.* For some schema still in the knowledge base, we can compute this by doing a mark-sweep along the schema's precomputed reachable schemas, immediately abandoning any path which mentions a schema in a lobotomized section.

In practice, we do not do this exhaustively for every lobotomy; instead, we maintain a hash table which caches lookups for any given schema. If some caller wishes the reachability information for a schema, it retrieves the cached information if present; otherwise, it computes it as described above and caches the result.

1. E.g., a Symbolics MacIvory Model 3.

Obviously, this cache must be flushed if the reachability criteria (e.g., the extent of the lobotomy) changes.

B.4 Computing paths from one arbitrary schema to another

The above cached reachability information now makes it possible to (relatively) quickly compute possible paths between any pair of schemas. In general, the number of possible paths between any given pair of schemas is quite large (perhaps millions in large runs); it would be helpful if we could quickly find just one path that is very likely to be the "best" we could do, especially since we must find such a path for *every* possible pair of schemas (e.g., hundreds of millions or billions of possible paths if looking at the full cross product).

Note that this is subtly different from the description of chaining (Section 4.3.2.4.1, on page 74) and path metrics (Section 4.3.2.4.2, on page 75) in the description of planning in Section 4.3.2.4. There, we were talking about picking a best path from amongst the thousands of possible paths from pairs of schemas the INITIAL and FINAL sets. Here, we are talking about picking a best path from amongst the millions of possible paths between some arbitrary pair of schemas. To put it another way, in Section 4.3.1.1, we were given (relatively) small sets of schemas—perhaps 50 to 100 schemas in each of INITIAL and FINAL. We had to find *one* path from *some* schema in INITIAL to *some* schema in FINAL; we were free to choose both ends of the path, plus the path itself, subject to the constraints of which schemas were in which set and the dictates of the path metric. Here, must find, *for each possible pair of schemas in the entire knowledge base*, some "best" path connecting that pair.

Thus, the problem here is as follows. We must compute, for the full cross product of every schema (call it A) to every other schema (call it B), what path to cache for the route

from A to B. Several approaches to the problem of computing a path from A to B are described below; any such approach would then have to be run several thousand or more times to generate the complete set of paths (e.g., a 3600-schema path could conceivably have 3600^2 or about 13 million paths; in practice, the graph is never that fully connected).

There are several approaches which are clear losers, as can be predicted from the AI literature on search problems (see, e.g., [Norvig 92], to pick a nice example). Depth-first search, for example, tends to favor long paths, which is a disaster in this system: the longer the path, the lower its expected reliability. A typical search in a large run might lead to a path several *hundred* actions long, when the "best" path (determined, say, by the algorithm described in Case 4 in Section 4.3.2.4.2, on page 75) might be only one or two actions.

Using A* search is an attractive possibility. Unfortunately, it is far too slow. Because we do not have a heuristic function available to guide the search, it takes far too long[2] to search the space, and conses enormous amounts intermediate garbage (in the form of path data structures and lists of them) in the process.

It turns out that using iterative deepening is in fact a very efficient way to generate short, useful paths in this case. This is where the cached reachability information becomes critically important: if we are guaranteed that *some* path exists between two schemas, then iterative deepening is guaranteed to find it. (And, in practice, the path found is generally half a dozen actions or less.) However, if *no* such path exists, then iterative deepening will run forever. Specifying an arbitrary upper limit on the length of a path may unnecessarily deprive us of a good path that is, e.g, just one action past our limit, but using reachability information, we may confidently set the upper limit at (effectively) infinity without worrying about failing to terminate.

2. Minutes to an hour or more on the aforementioned processor.

Using iterative deepening in this way, we stop computing the path between any given pair of schemas when we find the *first* one. Because of the way in which iterative deepening works, this path is guaranteed to be as short or shorter than any other possible path (if it were not, we would have found some different path at a shallower search depth). Rather than enumerate all such possible paths of this length, we simply assume that the first one we find is "good enough." In practice, this is perfectly reasonable behavior.

B.5 Computing a path between the INITIAL and FINAL sets

Given some combination of evaluation parameters as described in Section 4.3.2.9, on page 84 (e.g., the knowledge base of schemas to be evaluated, the extent of any lobotomies, and the path metric in use), we can build a cache of known-best paths from any pair of schemas (one in INITIAL, one in FINAL) that we have investigated. (This cache must be flushed if the evaluation parameters change.)

This works as follows. To recap the discussion in Section 4.3.2.5.4, on page 81, in the naive (e.g., no-cache) case, when planning a path to reach a goal, we must compute, for each possible pair of schemas in INITIAL and FINAL, the path from the selected schema (call it A) in INITIAL to the selected schema in FINAL (call it B). (The computation is performed, as it was in Section B.4 above, in an iterative depth-first manner.) Once we have computed all such paths, we then run the path metric on each such path, and pick the path with the best merit.

The iterative depth-first computation of the path from A to B will never change, however, as long as the evaluation parameters remain constant. We can therefore cache this work, so long as we are careful to flush the cache if these parameters change. Hence, the revised algorithm, before computing the path from A to B using iterative deepening, instead

checks the cache, and uses the cached path if it exists. Otherwise, it computes the path, and caches it.[3]

Using the cache in this way vastly speeds up computation of candidate paths. When the evaluation parameters are first changed (and the cache therefore flushed), the very first attempt at path planning generates several thousand to several tens of thousands of cached plans between different pairs of schemas. The next attempt at planning (after an action has been taken) generates a much smaller number of new entries in the cache, since many schemas that were in INITIAL and FINAL in the previous step may still be there (since only a few sensory bits in the world are likely to change at any given step). Empirically, the cache tends to grow asymptotically to about 5% of the total number of possible cross-products from all schemas to all other schemas, and does so in a small number (10 to 20) of actions executed. This is true because many schemas never wind up in INITIAL or FINAL in any given evaluation.

Hence, by caching the results of path lookup, we eliminate redundant calls to the iterative deepening algorithm. By computing the path between any given pair of schemas only on demand, instead of computing the entire cross-product, we decrease the required size of the cache and the effort of computing all those paths by a factor of 20 or so.

B.6 The promise of randomized algorithms

Despite the care with which the algorithms described above attempt to avoid combinatorial explosion and minimize redundant computation, they are still just barely in the bounds of reasonability for runs of several thousand schemas. On current hardware, generating the complete set of cached reachable schemas for a run of 3000-4000 schemas takes

3. The cache is implemented as a simple hash table, keyed by a number unique to each possible ordered pair of schema numbers. (For example, if x and y are the two schema numbers, a possible hash key could be $xn+y$, so long as $n>y$.) Keying by a number eliminates consing and allows use of an EQL hash table, which is extremely quick.\

most of an hour. Generating the majority of the cached INITIAL/FINAL paths for a run of that size can easily take another hour or more. Once several possible goals have been run in the same evaluation cycle, the results of the caching start to pay off, but planning is still somewhat slow, averaging a few seconds to half a minute or more (depending on the novelty of the situation, e.g, how many initial/final pairs were already cached) per plan generated.

It seems clear that, short of parallel implementation (which can at least pull one degree out of a polynomial blowup, at least until one runs out of processors), a better solution may to make use of *randomized algorithms* such as so-called *Monte Carlo* algorithms. This was not investigated in this research, but the significant speedups that randomized algorithms can offer in quickly finding a close-to-optimal path is probably well worth the small probability of not finding the truly optimal path. Given the somewhat ad-hoc nature of both the existing path metrics and the random nature of exactly which schemas have been created at any point in time, it is unlikely that a well-written randomized algorithm would noticeably degrade accuracy in path formation, and is likely to be orders of magnitude faster. This is an area deserving of future research.

Bibliography

D W Aha, D Kibler, and M K Albert, "Instance-Based Learning Algorithms," *Machine Learning 6*, pp. 37-66, 1991.

D Alan Allport, "Attention and Control: Have We Been Asking the Wrong Questions? A Critical Review of Twenty-Five Years," *Attention and Performance 14*, D E Meyer and S Kornblum (eds), Cambridge, MA: MIT Press.

Yiannis Aloimonos, editor, *Active Vision*, Lawrence Erlbaum Associates, Inc., 1993.

R A Andersen, "The Role of the Inferior Parietal Lobule in Spatial Perception and Visual-Motor Integration," *Handbook of Physiology: The Nervous System, Volume 5, Higher Functions of the Brain*, F Plum, V B Mountcastle, and S R Geiger (eds), Bethesda, MD: American Physiological Society, 1987.

R A Andersen, "Visual and Eye Movement Functions of the Posterior Parietal Cortex," *Annual Review of Neuroscience 12*, pp. 377-403, 1989.

Bruce Blumberg, "Action-Selection in Hamsterdam: Lessons from Ethology," *Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Brighton, England, 1994.

Lashon Booker, "Classifier Systems that Learn Internal World Models," *Machine Learning Journal*, Volume 1, Number 2,3, 1988.

Donald E Broadbent, *Perception and Communication*, Pergamon Press, London & NY, 1958.

Donald E Broadbent, *Decision and Stress,* Academic Press, London, 1971.

Rodney Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation, RA-2*, pp. 14-23, April 1986.

Bruce G Buchanan and Edward A Feigenbaum, Forward, *Knowledge Based Systems in Artificial Intelligence*, Randall Davis and Douglas B Lenat (eds), McGraw-Hill, NY, 1982.

B E Butler and A Currie, "On the Nature of Perceptual Limits in Vision: A New Look at Lateral Masking," *Psychological Research 48*, pp 201-209, 1986.

David Chapman, *Vision, Instruction, and Action*, MIT TR-1204 (doctoral thesis), April 1990.

Damaris Christensen, "A Bee and How It Sees, That is the Question," *Science News*, 145:11 (March 12 1994), p.166.

Michael T Cox and Ashwin Ram, "An Explicit Representation of Forgetting," *Proceedings of the Sixth International Conferences on Systems Research, Informatics and Cybernetics*, pp. 115-120, Baden-Baden, Germany, August 1992.

Michael T Cox and Ashwin Ram, "Choosing Learning Strategies to Achieve Learning Goals," to appear in *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, Marie desJardins and Ashram Ram (eds), Menlo Park, CA, AAAI Press, 1994.

O Creutzfeldt, "Multiple Visual Areas: Multiple Sensorimotor Links," *Models of Visual Cortex*, D Rose and V G Dobson (eds), pp. 54-61, New York: John Wiley, 1985.

Trevor Darrell, Pattie Maes, Bruce Blumberg, and Sandy Pentland, *Situated Vision and Behavior for Interactive Environments*, Technical Note #261, 1994, MIT Media Lab Perceptual Computing Group, Massachusetts Institute of Technology.

Marie desJardins, "Goal-Directed Learning: A Decision-Theoretic Model for Deciding What to Learn Next," *Proceedings of the ML-92 Workshop on Machine Discovery*, pp. 147-151, Ninth International Machine Learning Conference, University of Aberdeen, Scotland.

Daniel C Dennett and Marcel Kinsbourne, "Time and the Observer: The Where and When of Consciousness in the Brain," *Behavioral and Brain Sciences 15*, pp. 183-247, 1992.

T G Dietterich and R S Michalski, "Inductive Learning of Structural Description: Evaluation Criteria and Comparative Review of Selected Method," *Artificial Intelligence 16*, pp. 257-294, 1981.

E A DeYoe and D C Van Essen, "Concurrent Processing Streams in Monkey Visual Cortex," *Trends in Neurosciences 11*, pp. 219-226, 1988.

Gary Drescher, *Made Up Minds: A Constructivist Approach to Artificial Intelligence*, MIT Press, 1991.

Gary Drescher, personal communication, July 6, 1993.

Gary Drescher, personal communication, May 3, 1994.

J Driver and S P Tipper, "On the Nonselectivity of 'Selective Seeing:' Contracts Between Interfence and Priming in Selective Attention," *Journal of Experimental Psychology: Human Perception and Performance 15*, pp. 304-314, 1989.

R Ellis, D Alan Allport, G W Humphreys, and J Collis, "Varieties of Object Constancy," *Quarterly Journal of Experimental Psychology 41A*, pp. 775-796, 1989.

O Etzioni, "Hypothesis Filtering: A Practical Approach to Reliable Learning," *Proceedings of the Fifth International Conference on Machine Learning*, pp. 416-429, Morgan Kaufmann, Ann Arbor, MI, 1988.

J A Feldman, "Four Frames Suffice: A Provisional Model of Vision and Space," *Behavioral and Brain Sciences 8*, pp. 265-289, 1985.

Daniel J Felleman and David C Van Essen, "Distributed Hierarchical Processing in the Primate Cerebral Cortex," *Cerebral Cortex*, Jan/Feb 1991.

Leonard Foner, "Paying Attention to What's Important: Using Focus of Attention to Improve Unsupervised Learning," *Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, Brighton, England, 1994.

H Gallare and J Minker, *Logic and Data Bases*, Plenum, NY, 1978.

J Garcia, B K McGowan, and K F Green, "Biological Constraints on Conditioning," *Biological Boundaries of Learning*, edited by M E P Seligman and J L Hager, Appleton-Century-Crofts, New York, 1972, pp. 21-43.

J H Gennari, "Focused Concept Formation," *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 379-382, Morgan Kaufmann, Ithaca, NY, 1989.

A H C Van der Heijden, *Selective Attention in Vision*, Routledge, Chapman, and Hall, Inc, NY, NY, 1992.

John H Holland, "Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Appled to Parallel Rule-Based Systems," *Machine Learning: An Artificial Inelligence Approach, Volume 2*, R S Michalski, J G Carbonell, and T M Mitchell (eds), Morgan Kaufmann, Los Altos, CA, 1986.

G E Hinton and L M Parsons, "Scene-Based and Viewer-Centred Representations for Comparing Shapes," *Cognition 30*, pp. 1-35, 1988.

Lawrence Hunter, "Planning to Discover: Applications of Goal-Driven Learning to Molecular Biology," MIT AI Lab Revolving Seminar Series, April 21, 1994.

G A Iba, "A Heuristic Approach to the Discovery of Macro-Operators," *Machine Learning 3*, pp. 285-317, 1989.

J C Johnston and H Pashler, "Close Binding of Identity and Location in Visual Feature Perception," *Journal of Experimental Psychology: Human Perception and Performance 16,* pp. 843-856, 1990.

W A Johnston and V J Dark, "Selective Attention," *Annual review of Psychology 37,* pp. 43-75, 1986.

Bela Julesz, "A Brief Outline of the Texton Theory of Human Vision," *Trends in Neuroscience*, pp. 41-45, February 1984.

Leslie Pack Kaelbling, *Learning in Embedded Systems*, MIT Press, 1993.

Leslie Pack Kaelbling and David Chapman, "Learning from Delayed Reinforcement in a Complex Domain," Teleos TR-90-11, December 1990.

D Kahneman, *Attention and Effort*, Englewood Cliffs, NJ: Prentice-Hall, 1973.

D Kahneman and Anne M Treisman, "Changing Views of Attention and Automaticity," *Varieties of Attention*, R Parasuraman and D R Davies (eds), pp. 29-61, New York: Academic Press, 1984.

M R Keller, "Concept Learning in Context," *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 482-487, Morgan Kaufmann, Irvine, CA.

R A Kinchla, "The Measurement of Attention," *Attention and Performance 8*, R S Nickerson (ed), Lawrence Erlbaum Associates, NJ, 1980.

Christof Koch and Shimon Ullman, "Selecting One Among the Many: A Simple Network Implementing Shifts in Selective Visual Attention," *Human Neurobiology 4*, pp. 219-227, 1985. Also MIT AI Memo 770 and CBIP Paper 003, January 1984.

David Leake and Ashwin Ram, "Goal-Driven Learning: Fundamental Issues and Symposium Report," Technical Report 85, Cognitive Science Program, Indiana University, Bloomington, Indiana, January 1993.

A R Luria, E N Pravdina-Vinarskaya, and A L Yarbuss, "Disorders of Ocular Movement in a Case of Simultanagnosia," *Brain 86*, pp. 219-228, 1963.

Pattie Maes, "ALIVE: An Artificial Life Interactive Video Environment," *Visual Proceedings, The Art and Interdisciplinary Programs of Siggraph 1993.*

Pattie Maes, "Learning Behavior Networks from Experience," *Toward a Practice of Autonomous Systems, Proceedings of the First European Conference on Artificial Life*, edited by F J Varela and P Bourgine, MIT Press/Bradford Books, 1992.

Pattie Maes, "Modeling Adaptive Autonomous Agents," *Artificial Life Journal, Volume 1, Numbers 1/2* (Fall 1993/Winter 1994), Christopher G Langton (ed), MIT Press, Cambridge, MA, 1994.

Scott Markovitch and Paul D Scott, "The Role of Forgetting in Learning," *Proceedings of the Fifth International Conference on Machine Learning*, pp. 459-465, Morgan Kaufmann, Ann Arbor, MI, 1988.

Scott Markovitch and Paul D Scott, "Post-PROLOG: Structured Partially Ordered PROLOG," TR CMI-89-018, Center for Machine Intelligence, Ann Arbor, MI, 1989.

David Marr, *Vision*, San Francisco: Freeman, 1982.

Marvin Minsky, *The Society of Mind*, Simon and Shuster, 1986.

T M Mitchell, "Generalization as Search," *Artificial Intelligence 18*, pp. 203-226, 1982.

T M Mitchell, P E Utgoff, and R Banerji, "Learning by Experimentation: Acquiring and Refining Problem-Solving Heuristics," *Machine Learning: An Artificial Intelligence Approach*, R S Michalski, J G Carbonell, and T M Mitchell (eds), Tioga, Palo Alto, CA, 1983.

R Mooney, "The Effect of Rule Use on the Utility of Explanation-Based Learning," *Proceedings of the Eleventh International Joint Conference for Artificial Intelligence,* pp. 725-730, Morgan Kaufmann, Detroit, MI, 1989.

Andrew Moore and Christopher G Atkeson, "Prioritized Sweeping: Reinforcement Learning with Less Data and Less Time," *Machine Learning 13:1*, pp. 103ff, October 1993.

S Minton, *Learning Search Control Knowledge: An Explanation-Based Approach*, Kluwer, Boston, MA, 1988.

L A Morrow and G Ratcliff, "The Disengagement of Covert Attention and the Neglect Syndrome," *Psychobiology 3*, 261-269.

D Mumford, "On the Computational Architecture of the Neocortex: I. The Role of the Thalamo-Cortical Loop," *Biological Cybernetics 65*, pp. 135-145, Springer-Verlag, 1991.

Peter Norvig, *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*, Morgan Kaufmann, San Mateo, CA, 1992.

Michael I Posner, Charles R R Snyder, and Briand J Davidson, "Attention and Detection of Signals," *Journal of Experimental Psychology: General*, 109:2, pp. 160-174, 1980.

Michael I Posner, A W Inhoff, F J Friedrich, and A Cohen, "Isolating Attentional Systems: A Cognitive-Anatomical Analysis," *Psychobiology 15*, pp. 107-121, 1987.

J R Quinlan, "Induction of Decision Trees," *Machine Learning 1*, pp. 81-106, 1986.

Ashwin Ram, "Incremental Learning of Explanation Patterns and their Indices," *Proceedings of the Seventh International Conference on Machine Learning*, pp. 313-320, Austin, TX, June 1990a.

Ashwin Ram, "Knowledge Goals: A Theory of Interestingness," *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Cambridge, MA, August 1990b.

Ashwin Ram and Michael T Cox, "Introspective Reasoning Using Meta-Explanations for Multistrategy Learning," to appear in *Machine Learning: A Multistrategy Approach, Volume 4*, R S Michalski and G Tecuci (eds), Morgan Kaufman, San Mateo, CA, 1994, and also as Georgia Institute of Technology TR GIT-CC-92/19, Atlanta, Georgia, 1992.

Ashwin Ram and Lawrence Hunter, "The Use of Explicit Goals for Knowledge to Guide Inference and Learning," *Applied Intelligence* 2:1, pp. 47-73, 1992, and also as Georgia Institute of Technology TR GIT-CC-92/04, Atlanta, Georgia, 1992.

Ashwin Ram and David Leake, "A Framework for Goal-Driven Learning," to appear in *Proceedings of the AAAI Spring Symposium on Goal-Driven Learning*, Stanford, CA, March 1994.

Ashwin Ram, S Narayanan, and Michael T Cox, "Learning to Troubleshoot: Multistrategy Learning of Diagnostic Knowledge for a Real-world Problem Solving Task," Georgia Institute of Technology TR GIT-CC-93/67, Atlanta, Georgia, 1993.

Robert Ramstad, *A Constructivist Approach to Artificial Intelligence Reexamined (*MIT combined Bachelor's and Master's thesis, 1992).

Arthur L Samuels, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal 3*, pp. 211-229, 1959.

J F Soechting, S I H Tillery, and M Flanders, "Transformation from Head- to Shoulder-Centered Representation of Target Direction in Arm Movements," *Journal of Cognitive Neuroscience 2*, pp. 32-43, 1990.

G Sperling, S A Wurst, and Z Lu, "Using Repetition Detection to Define and Localize the Processes of Selective Attention," *Attention and Performance 14*, D E Meyer and S Kornblum (eds), MIT Press, Cambridge, MA, 1990.

Bob Sproull, "Counterflow Pipeline Processor Architecture," MIT AI Lab seminar , March 14, 1994.

Craig Stanfill and David Waltz, "Toward Memory-Based Reasoning," *Communications of the ACM*, 29:12, pp. 1213-1228, December 1986.

Rich S Sutton, "Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming," in *Proceedings of the Seventh International Conference in Machine Learning*, Austin, TX, June 1990.

Ming Tan, "Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics," *Machine Learning 13*, 1993.

Sabastian B Thrun, "Efficient Exploration in Reinforcement Learning," *Technical report CMU-CS-92-102*, Carnegie Mellon University, Pittsburgh, PA, January 1992.

Sebastian B Thrun, "The Role of Exploration in Learning Control," to appear in *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, David A White and Donald A Sofge, Van Nostrand Reinhold, Florence, KY, 1994.

Anne M Treisman, "Strategies and Models of Selective Attention," *Psychological Review 76*, pp. 282-299, 1969.

Anne M Treisman, "Features and Objects: The Fourteenth Bartlett Memorial Lecture," *Quarterly Journal of Experimental Psychology 40A*, pp. 201-237, 1988.

Anne M Treisman and Garry Gelade, "A Feature Integration Theory of Attention," *Cognitive Psychology 12*, pp. 97-136, 1980.

Anne M Treisman and Stephen Gormican, "Feature Analysis in Early Vision: Evidence From Search Asymmetries," *Psychological Review* , 95:1, pp. 15-48, 1988.

Shimon Ullman, "Sequence-Seeking and Counter Streams: A Model for Information Processing in the Cortex," MIT AI Lab AI Memo 1311, 1991.

L G Ungerleider and M Mishkin, "Two Cortical Visual Systems," *Analysis of Visual Behavior*, D J Ingle, M A Goodale, and R J W Mansfield (eds), Cambridge, MA: MIT Press, 1982.

Alan R White, *Attention*, Blackwell: Oxford, England, 1964.

Steven D Whitehead, "Complexity and Cooperation in Q-Learning," *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 363-367, 1991.

Steven D Whitehead, "A Study of Cooperative Mechanisms for Faster Reinforcement Learning," *Technical Report 365*, University of Rochester, Computer Science Department, Rochester, NY, March 1991.

Steven D Whitehead and Dana Ballard, "Active Perception and Reinforcement Learning," submitted to the Seventh International Conference on Machine Learning, Austin, TX, June 1990.

Stewart W. Wilson, "Knowledge Growth in an Artificial Animal," *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, edited by Greffenstette, Lawrence Erlbaum Associates, 1985.

John Woodfill and Ramin Zabih, "An Architecture for Action with Selective Attention," submitted to AAAI-90.