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 72, 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 71, 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 79, 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 94, 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 48.

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 76, 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 71.

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 79.

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 89).

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 48.

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 48. In particular, the iterators that determine which schemas and sensory items are attended to (see Section 3.2.3, on page 52) 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 52; 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 77. This FSM was designed for the infant/eyehand scenario; the Ham-

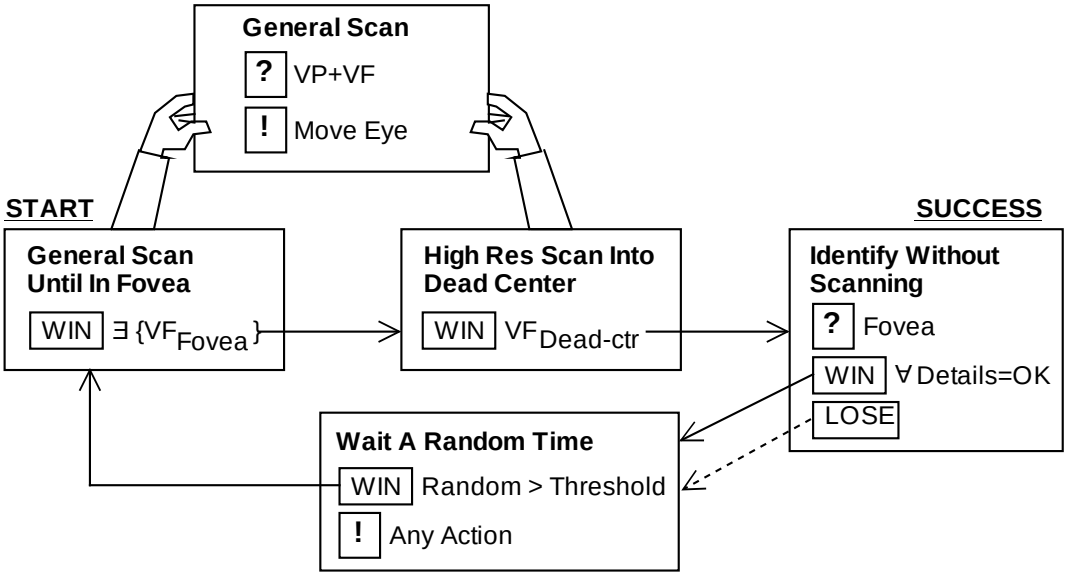

Figure 9: A typical set of goals

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 91, 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 94, 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 57, 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 72 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 87, 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 82.) 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 29, 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 91, 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 91).

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 92, 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 89) 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 96, 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 79, 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 77. 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 72—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 63, with additional results for a typical goal run (in this case, the goal demonstrated in Figure 9, on page 77). 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 72 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 83, 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 79, 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 77, 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 97, 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 101, and Figure 14, on page 102.[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 88). These and the other table headings correspond to those described in the discussion of scorecards in Section 4.3.2.9, on page 92.

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

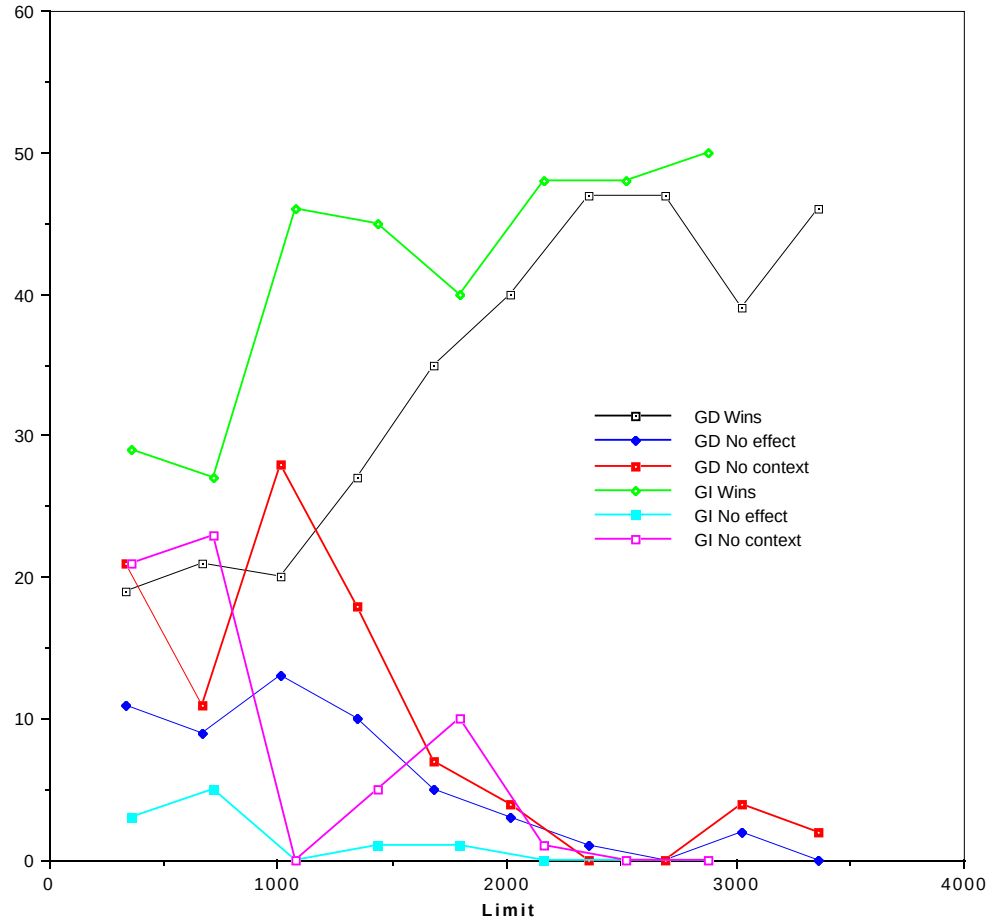

Figure 15: Goal-independent and goal-dependent performance compared

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 108, 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 83) 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 108.[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.