In *Working Notes of the Symposium on Representing Mental States and Mechanisms, AAAI-95 Spring Symposium Series*, pp. 15-21.

A Representation and Learning Mechanisms for Mental States

Paul Cohen, Marc Atkin, Tim Oates and Dawn Gregory

Computer Science Technical Report 95-53

Experimental Knowledge Systems Laboratory
Computer Science Department, Box 34610
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4610

Abstract

We want to build an agent that plans by imagining sequences of future states. Subjectively, these states seem very rich and detailed. Providing an agent with sufficiently rich knowledge about its world is an impediment to studying this kind of planning, so we have developed mechanisms for an agent to learn about its world. One mechanism learns dependencies between synchronous "snapshots" of the world; the other learns about processes and their relationships.

This research is supported by ARPA/Rome Laboratory under contract F30602-93-C-0100.

1 A Motivating Example

Imagine an old kitchen cabinet, recently removed from a kitchen wall, six feet long, with doors but no back, nails sticking out of odd places, splintered where the crowbar did its work. This cabinet rests on the basement floor, but you want to attach it to the basement wall. It weighs about 50 lbs and it's very cumbersome. Your first thought is to attach a batten to the back of the cabinet along its length, then drill screw holes in the basement wall, then drill through the batten at locations that correspond to the holes in the wall. You intend to lift the cabinet four feet off the ground, register the batten holes with the screw holes, and screw the cabinet to the wall. Running through this plan in your mind, you realize it won't work, because you cannot hold a 50 lb., six-foot, structurally unsound cabinet four feet off the ground with one hand, while you screw it to the wall with the other. You need another person to help you, or you must build some sort of scaffolding to hold the cabinet in place. Suppose neither option is feasible. After thinking about it for a while, you suddenly come up with a new plan: Attach three or four L-brackets to the basement wall, hoist the cabinet onto the L-brackets, and then secure it to the wall. As you run through this plan mentally you recognize several hazards: the doors will swing and get in the way; the nails are dangerous and must be removed; you must not grasp the cabinet where the wood is splintered. Subjectively, each hazard seems to be "read" from a mental movie of sorts: You imagine hoisting the cabinet, having it lean slightly toward you, and a door swinging open and knocking your spectacles off your nose. You imagine holding the cabinet against the wall with your shoulder (now that its weight is supported by the L-brackets) leaving two hands free to drive in the screws, but then you realize that if you drop a screw, you can't bend down to pick it up, so you modify your plan and put a bunch of screws in your shirt pocket. Or if you don't have a pocket, you hold them in your teeth. You can almost feel the metal chinking against your teeth.

The most striking thing about this example is how much you think about, and how rich your mental images seem. Another characteristic of the example is the "functional plasticity" of its components: Your shoulder becomes something to brace against the cabinet; your mouth becomes something to hold screws; the cabinet doors become something that hit you in the face; the nails, which once functioned as fasteners, now tear your flesh. A third characteristic of the example is that once you have a skeletal plan (e.g., lift the cabinet onto the L-brackets and attach it to the wall) you seem to fill in the details by imagining executing the plan, by visualization and forward simulation. Indeed, this is how you discovered that the original plan wouldn't work. Perhaps crude plans can be generated by conventional, propositional AI algorithms, but checking a plan seems to require some ability to imagine or visualize oneself executing it.

Subjectively, the frame problem and problems of relevance don't seem to arise (McCarthy and Hayes, 1969). Suppose that in an attempt to have the screws near at hand, you place them on one of the shelves of the cabinet before you lift it. Where are the screws when the cabinet has been hoisted into place? Subjectively, in your mind's eyes and ears, you can see and hear the screws as they roll off the shelf and fall on the floor. Similarly, the relevance of swinging doors seems to emerge as you imagine hoisting the cabinet. Subjectively it isn't difficult to envision future states, nor are we troubled by the impossibility of knowing precisely how the world will look after an action. We know *enough* about processes such as hoisting cabinets to support planning by visualization. The focus of this paper is how we learn about processes.

Our purpose here is not to enter the debate about the nature and implementation of mental imagery (Computational Intelligence, 1994). Nor are we claiming that propositional AI planning is incompatible with the sort of imagining we just described. Instead we are asking what it would take for an AI planner to plan by forward simulation through very rich mental states; we want to know whether it would display functional plasticity, that is, whether it could determine the relevance of objects de novo as it imagines a plan unfolding. This paper does not answer these questions—it doesn't even present a planner—but it does describe how an agent can learn representations of environment states and rules to predict the "next" state given the "current" one.

2 Streams World

The first impediment to studying these questions empirically is building an agent that has rich mental states. To encode by hand everything that an agent must know to solve the kitchen cabinet problem by visualization would take months or years. Instead we decided that the agent should learn everything it needs to know about its environment through observation, freeing us from the task of constructing the agent's knowledge base by hand.

The Streams World is an abstract environment in which the sensory array can be arbitrarily large and the dependencies that hold among sensed states can be arbitrarily complex. The environment is a set of time series of tokens, called *streams*. At each time step, a new token is appended to the end of each stream and is made accessible to the agent's sensors. Previous tokens become inaccessible, although the agent might have stored some of them. Suppose we look at three streams:

```
Stream 1:  G A B A G C D A A B G A C D B ...
Stream 2:  X W A X X W Q W Q W Q A A A X ...
Stream 3:  1 2 2 2 4 2 4 3 5 7 7 8 9 3 2 ...
           . . .
```

By convention, the leftmost column of tokens is the state of the streams "now." A *multitoken* is a vertical slice; for example, the "current" multitoken is <G X 1> and the most recent previous multitoken is <A W 2>. Multitokens can include wildcards; for example, the multitoken <G X *> happens three times in the history above.

We constructed the Streams World simulator so that the value of a token in a stream (say, A in stream 1) can depend probabilistically on earlier tokens in the stream and on tokens in other streams.

3 Learning Dependencies among Multitokens

A large part of being able to visualize future states is to construct temporally-ordered associations between states. Multitokens are representations of states, or classes of states when the multitokens include wildcards. As a prelude to planning, then, our agent must learn multitoken transition rules. Given a multitoken or a remembered sequence of multitokens, the agent must predict the next multitoken or sequence of multitokens. More generally, as we will see, the agent needs to learn about processes that extend over time. Learning about multitoken sequences is not sufficient, but is a good place to begin.

We have developed two algorithms to learn such rules. One runs incrementally, extending its rules as new multitokens occur. The other stores a long history of multitokens and replays it in batch mode to learn rules. The former is more psychologically plausible, but it is less well developed than the batch algorithm, which we describe here. Both algorithms learn dependencies between individual multitokens, that is, neither learns dependencies between sequences of multitokens. We will return to this issue later.

The multi-stream dependency detection algorithm (MSDD) is an extension of Adele Howe's algorithm for finding dependencies between tokens in a single stream (Oates, Gregory and Cohen, 1995; Howe and Cohen, 1995). We will describe Howe's algorithm first.

Let (p, s, δ) denote a dependency. Each dependency *rule* says that when the precursor token, p, occurs at time step i in the stream, the successor token, s, will occur at time step $i+\delta$ in the stream with some probability. When this probability is high, the dependency is strong.

Consider the stream ACBABACCBAABACBBACBA. Of all 19 pairs of tokens at lag 1 (e.g. AC, CB, BA, ...) 7 pairs have B as the precursor; 6 of these have A as the successor, and one has something other than A (denoted $\overline{\text{A}}$), as the successor. The following contingency table represents this information:

$Table$(B,A,1) =

	A	$\overline{\text{A}}$	*total*
B	6	1	7
$\overline{\text{B}}$	1	11	12
total	7	12	19

It appears that A depends strongly on B because it almost always follows B and almost never follows $\overline{\text{B}}$. We can determine the significance of each dependency by computing a G statistic for its contingency table. The table shown above has a G value of 12.38, significant at the .001 level, so we reject the null hypothesis that A and B are independent and conclude that (B,A,1) is a real dependency.

Now we can generalize Howe's method to multiple streams. For a set of n streams, all multitokens will have the form $< x_1, \ldots, x_n >$, where x_j indicates the value in stream j. Consider the following streams:

AC**BABA**CCBAABAC**BA**C**BA**C
BA**CACA**BACBABAB**CA**B**CA**B

The dependency (<B,C>,<A,A>,1) indicated in boldface is significant at the .01 level with a G value of 7.21. The corresponding contingency table is:

$Table$(B,A,1) =

	<A,A>	$\overline{\text{<A,A>}}$	*total*
<B,C>	4	1	5
$\overline{\text{<B,C>}}$	2	12	14
total	6	13	19

We now have both syntax and semantics for multistream dependencies. Syntactically, a dependency

is a triple containing two multitokens (a precursor and a successor) and an integer (the lag). Dependencies can also be expressed in the form $x \rightarrow_\delta y$ where x and y are events. Semantically, this says the occurrence of x is indicative of or predicts the occurrence of y, δ time steps in the future.

The problem of finding significant two-item dependencies is framed in terms of search. A node in the search space consists of a precursor/successor pair, a predictive rule. The goal is to find predictive rules that are "good" in the sense that they apply often and are accurate. The root of the search space is a pair of multitokens with wildcards in all n positions. The children of a node are generated by replacing (instantiating) a single wildcard in the parent, in either the precursor or successor, with a token that may appear in the appropriate stream. For example, the node $<$ A,* $>\rightarrow<$ *,X $>$ has both $<$ A,Y $>\rightarrow<$ *,X $>$ and $<$ A,* $>\rightarrow<$ B,X $>$ as children.

The rule corresponding to a node is always more specific than the rules of its ancestors and less specific than any of its descendants. This fact can be exploited in the search process by noting that as we move down any branch in the search space, the value in the top left cell of the contingency table (n_1) can only remain the same or get smaller. This leads to a powerful pruning heuristic. Since rules based on infrequently co-occurring pairs of multitokens (those with small n_1) are likely to be spurious, we can establish a minimum size for n_1 and prune the search space at any node for which n_1 falls below that cutoff. In practice, this heuristic dramatically reduces the size of the search space that needs to be considered.

Our implementation of MSDD is a *best first search* with a heuristic evaluation function that strikes a tunable balance between the expected number of hits and false positives for the predictive rules when they are applied to previously unseen data from the same source. We define *aggressiveness* as a parameter, $0 \leq a \leq 1$, that specifies the value assigned to hits relative to the cost associated with false positives. For a given node (rule) and its contingency table, let n_1 be the size of the top left cell, let n_2 be the size of the top right cell, and let t_S be the number of non-wildcards in the successor multitoken. The value assigned to each node in the search space is $S = t_S(an_1 - (1-a)n_2)$. High values of aggressiveness favor large n_1 and thus maximize hits without regard to false positives. Low aggressiveness favors small n_2 and thus minimizes false positives with a potential loss of hits. Since the size of the search space is enormous, we typically impose a limit on the number of nodes expanded. The output of the search is simply a list of the nodes, and thus predictive rules, generated.

4 Evaluating MSDD

Because Streams World is an abstract environment, we can control the number of streams, the number of token values in each stream, the number of unique multitokens, and the complexity of the probabilistic relationships among multitokens. Thus, we can present an agent with an arbitrarily rich environment from which to construct rules governing multitoken transitions.

In one experiment (Oates et al.,1995) we generated streams that were for the most part random but contained a small number of probabilistic dependencies between multitokens. Our independent variables were the number of streams (5, 10, 20); the length of streams (100, 1000, 5000); the number of unique tokens in each stream (5, 10, 20). In each condition, streams were seeded with dependencies. The probability that any predecessor would appear was .1, .2 or .3, and the probability that the successor would follow the predecessor was .1, .5 or .9. Predecessor and successor multitokens could contain 1, 3, or 5 non-wildcard tokens, although these parameters were varied factorially, so sometimes a highly specific predecessor predicted a general successor, or vice versa. We ran MSDD under all these conditions, collected the predictive rules it generated, and used them to predict multitokens in new streams that had the same probabilistic dependency structure as the training streams. In general, every multitoken in the test streams was matched by more than one predictive rule, so we also tested a number of heuristics for selecting rules, that is, deciding which prediction to make.

In general, the only limitation on MSDD's accuracy is the probability that a successor follows a predecessor. In particular, the probability of seeing a predecessor and the degree of specificity of the predecessor and successor mutitokens do not affect accuracy. MSDD's performance actually improves as the number of unique tokens in streams increases. The length of streams has very little impact on MSDD's accuracy. Although MSDD generates many predictive rules, most of which are spurious, it is easy to separate good rules from bad. In the previous experiments, we discarded those with $n_1 < 5$ and sorted the remaining rules by their G statistics. The following experiment used a slightly different criterion.

Because MSDD rules contain pairs of multitokens,

we decided to apply MSDD to classification problems in which the "precursor" is a feature vector and the "successor" is a classification. We ran the algorithm on a dozen datasets from the Irvine Machine Learning repository. Most of these datasets were chosen from a set of 13 identified by Zheng (1994) as a minimal representative set. We compared MSDD's performance with other published results for each dataset. On ten datasets for which multiple results have been published, MSDD performance exceeds that of half the reported results on six datasets. In only one case did it perform badly (the soybean dataset), and often it performed extremely well (Oates, Gregory and Cohen, 1995).

5 Fluents vs. Multitokens

With multitoken transition rules for the abstract Streams World environment, an agent can imagine possible future multitokens by chaining through its rules. But when we consider how real environments might be encoded in streams we run into a problem. In real environments, states persist for different amounts of time, or change over time. Suppose an environment includes a fancy electronic rattle that emits a pair of tones, high-low-high-low, when it's shaken. We want an agent to learn that the rattle makes noise when shaken. The natural way to think of this is that a process, shaking, is associated with another process, noise. It doesn't matter how long the agent shakes the rattle; what matters is the relationship between the onset of shaking and the onset of noise, and, at the other end, the terminations of shaking and noise. If you observe that Y starts shortly after X starts, and ends shortly after X ends, then you'd have no trouble associating the two events. MSDD has trouble because it's busy searching for associations between multitokens at each time step—exactly the wrong approach if you want to learn associations between persistent phenomena. Suppose AAABBB in stream 1, below, represents the high-low tone of the rattle; PO in stream 2 represents a forward arm motion followed by a backward one; and R in stream 3 represents grasping the rattle. We want the agent to learn that waving (POPO...) while grasping the rattle (R...) results in noise (AAABBBAAABBB...). Instead, MSDD will learn a lot of rules such as $< \text{A,P,R} > \rightarrow < \text{A,O,R} >$ and $< \text{A,*,*} > \rightarrow < \text{A,*,*} >$. It will not learn that AAABBB in stream 1 is a single instance of a repeating process, or that $(\text{AAABBB})^+$ represents an unchanging pattern. (The plus sign denotes repetition.)

```
Stream 1:  A A A B B B A A A B B B C D B C ...
Stream 2:  P O P O P O P O P O P O P O V X ...
Stream 3:  R R R R R R R R R R R R R R R R ...
```

Borrowing from McCarthy and Hayes (1969), we call such patterns *fluents*. Informally, a fluent is something that doesn't change over time. The components of the $(\text{AAABBB})^+$ fluent change, but their pattern does not. Fluents are analogous to the Gibsonian notion of perceptual invariant—something that remains constant amidst change, like the derivative of a function (Gibson, 1979). The durations of fluents vary, whereas the duration of a multitoken is just one time step. We want agents to learn to characterize the current state by the fluents that are currently active, not by the current multitoken. Although the current state in the example, above, is the multitoken A,P,R, we want an agent to characterize it as "noise, waving, grasping," if these are the active fluents.

The question of how fluents become active, indeed, what "active" means, must be addressed in a discussion of mental states. The *scope* of a fluent is the set of streams in which activity can *trigger* a fluent; for instance, stream 1 is the scope of the fluent $(\text{AAABBB})^+$. When a fluent is triggered, it starts to make predictions about the contents of the streams in its scope, and if these predictions are correct for a number of time steps, then the fluent becomes *active*. Conversely, an active fluent that makes bad predictions for some period will become inactive. Some but not all active fluents are *attended*. The distinction arises because we want an agent to not attend to something (e.g., the lighting in a room) but still notice when it changes. Attended means available for mental work; in particular, the system we describe below learns associations only among attended fluents. We call attended and active fluents, but not triggered fluents, the constituents of mental state.

This activation scheme has many variants, none of which are implemented, currently. First, it seems plausible that active fluents should influence which fluents become active. This might be done by "boosting" triggered fluents that are associated with currently active ones. An even simpler scheme involves activating composite fluents and their constituents whenever one constituent becomes active. For example, if the fluent A^+ becomes active, so will $(\text{AAABBB})^+$ and consequently, B^+. Second, some fluents are more important than others because they are associated with extraordinary pain or pleasure. It seems plausible that they might become active as soon as they are triggered. Third, some events happen very quickly, so there isn't time for a triggered fluent to prove itself by making accurate predictions

for several time steps. If the phone rings while you are reading this paper, it becomes part of your mental state very quickly; you don't require the triggered phone-ringing fluent to predict several rings before you activate the fluent. Fourth, the ambiguity of sensory experience is such that several fluents with identical or overlapping scope might be triggered. We require some methods to assess which of these alternative interpretations of the world should become active.

Note that mental states correspond to external world states in the sense that fluents are triggered by perceived world states and become (and remain) active if they can predict world states. Fluents are interpretations of sensory data in streams, but they remain closely associated with sensory data. Thus the fluent $(\text{AAABBB})^+$ tells the agent that two sensory experiences follow each other repeatedly. In this sense, fluents provide the sort of abstraction of sensory information that Jean Mandler argues is essential for infant conceptual and language development (Mandler, 1992). According to Mandler, the basis for conceptual development is sensory regularities; for example, babies can distinguish animate and inanimate motion very early, so they have a perceptual basis for organizing their worlds into two classes—things that move in an up-and-down, loping way, and things that move in a straight line. These classes become refined and abstract, and infants eventually learn names for them, but their genesis is in a distinction made at a perceptual level. Similarly, if an agent distinguishes at a perceptual level the alternating pattern $(\text{AAABBB})^+$ from the repeating pattern $(\text{CCC})^+$, then the alternating/repeating distinction is a basis for classification. It might not be a very helpful basis; the class of things that present alternating sequences of, say, sounds or colors might not be a *basic* class (in the sense Rosch uses the term). Whether of not the alternating/repeating distinction is helpful, it ought to be possible to build an agent to detect regularities in fluents (and relationships between fluents) that do provide a perceptual basis for basic categories.

Fluents predict themselves (as you would expect from a structure that persists over time) and they also are associated with other fluents. Thus $(\text{AAABBB})^+$, $(\text{PO})^+$ and R^+ might be associated. One is tempted to view fluents as "concepts" in a "semantic network," but they differ in one important respect: the links between fluents do not carry meaning, as the links between nodes do in a semantic network. Links between fluents indicate association, only. The semantics of fluents are provided by their grounding in streams; in particular, we describe an agent with affective streams in the following section. For this agent, the meaning of a fluent is the interest, pleasure or pain associated with the fluent.

6 Learning Fluents in Baby World

The Streams World offers flexibility and control, but it does not model any environment in particular. We have implemented another environment according to the same principles—tokens in streams are dependent and change over time—but this one is slightly more realistic. A simulated "baby" can see, hear, turn its head, wave its arms and perform other actions in an environment that includes rattles, a mobile, a bottle, another agent ("Mommy") and, most cruelly, a knife. Baby has access to external streams that represent sights, sounds and haptic sensations; and it has internal streams that register pleasure, pain, interest, hunger, and somatic sensations.[1] Dependencies between tokens and streams are encoded into the simulator; for example, when the mobile spins, Baby's pleasure and interest streams change, and when Baby grasps the knife, the pain stream changes. If Baby howls for a while, then Mommy will probably enter its field of view and Baby's pleasure stream will change. We await the day that Baby howls in order to bring Mommy and experience pleasure. Even this rudimentary planning is currently beyond Baby's grasp.

We have implemented a fluent-learning mechanism for Baby. It has not been evaluated, but we will describe how it works and show some examples of fluents it has learned.

Fluent learning has three aspects. First, to learn individual fluents, an agent must learn regularity in a subset of streams, that is, the agent must learn a fluent's scope, and what happens in the streams in the scope. Second, the agent must learn associations between fluents. Third, fluents (particularly scopes) might be revised. We have implemented the first and second type of fluent learning; the third depends on more sophisticated attentional mechanisms, as described below.

Because Baby is embedded in an ongoing environment, and has essentially unlimited opportunities to learn, it needn't extract all the structure from its

[1] We thank Matt Schmill for building the CLIM interface to Baby World. To obtain Baby World for your own use, send mail to dhart@cs.umass.edu.

streams. Instead, it can sample the streams, detecting snippets of structure, over time. Thus, the first kind of fluent learning involves repeated sampling of just one or two streams, at random, for a few time steps, looking for regularities. Suppose Baby samples stream 1, above. It will observe three A tones, followed by three B's. When something is repeated in a stream, a fluent is formed. Thus, Baby learns A^+ and B^+, two fluents, each with stream 1 as the scope. In stream 2, the pattern PO is repeated, so Baby learns PO^+. Baby also learns that A^+ precedes B^+, and the pattern A^+B^+ repeats, so it can form a larger fluent $(A^+B^+)^+$.

Baby doesn't require every occurrence of a fluent to last exactly the same number of time steps; for example, AAAB... and AAAAAAAB... are both occurrences of A^+ (terminating with B). Interestingly, Baby can easily learn the average length of a fluent by examining its contingency table. Suppose stream 1 contains

A A A A A B B B A A B B B A A A A A A B B.

This yields a contingency table for testing the significance of the fluent A+:

$$Table(A,A,1) = \begin{array}{cccc} & A & \overline{A} & total \\ A & 10 & 3 & 13 \\ \overline{A} & 2 & 5 & 7 \\ total & 12 & 8 & 20 \end{array}$$

For this table $G = 4.5$, a significant value, so we conclude that A follows A more than we'd expect by chance if occurrences of A were independent. Notice that the ratio of the first row total and the top-right cell, $13/3 = 4.33$, is the average length of the fluent A+. Thus Baby learns a fluent and its temporal extent. Given a fluent's duration, it's easy to imagine Baby testing the hypothesis that a particularly long or short version of the fluent differs significantly from the average (a t test would do it), and perhaps forming a subclass of "long A^+." (We have not yet implemented this idea.)

These simple fluents can now be associated. Obviously the space of combinations of fluents is very large, so Baby learns associations only between attended fluents, and only when they start or end at roughly the same time, or when one ends and another begins. Here are two examples:

```
START (& (+ TIREDNESS TIRED)(+ VOICE QUIET))
      (+ HUNGER SOMEWHAT-HUNGRY)
PREDICT (+ TIREDNESS TIRED)
        (+ VOICE CRYING)
```

In the first example, three fluents are associated. The TIREDNESS TIRED and VOICE QUIET fluents have already been associated to form the conjunctive fluent (& (+ TIREDNESS TIRED)(+ VOICE QUIET)) and now the conjunction is associated with (+ HUNGER SOMEWHAT-HUNGRY) because they start at roughly the same time. That is, Baby notices a change of state in the scope of the conjunctive fluent (i.e., the TIREDNESS and VOICE streams) at about the same time it observes a change in the HUNGER stream. In the second example, Baby notices that TIREDNESS TIRED precedes and thus predicts (+ VOICE CRYING).

The mechanics of association are by now familiar: We build a contingency table for each type of association between fluents X and Y. For instance, to test whether X and Y start at roughly the same time, we count the number of times X and Y start within 10 time steps of each other, and the number of times X is roughly coincident with something other than Y, and the coincidences of $\overline{X}$ and Y, and $\overline{X}$ and $\overline{Y}$. If the resulting G statistic is significant, we say X and Y start together (within 10 time steps) to a significant extent. The procedure is the same for other types of association, which include:

Stop. When X ends, Y ends.

Terminate. When X begins, Y ends.

Predict. When X ends, Y begins.

All these relationships are "rough" in the sense that X and Y must co-occur not exactly, but within a window of several time steps duration.

A third type of fluent learning corresponds to what Piaget called accommodation—the incremental modification of fluents through experience. We haven't implemented accommodation because it requires more sophisticated attentional mechanisms than Baby has now.

Attention and learning are tightly coupled in our scheme, because Baby learns only what it attends to, and its attention will be directed in part by what it knows. Currently, Baby attends to all active fluents, because we haven't implemented any means of focusing attention. It would be relatively easy to cobble something together and call it an attentional mechanism, but we are reluctant to do this because we think attention is important to learning, autonomous behavior and consciousness.

For starters, attention is related to one's *task* or *goals*, but Baby has neither. Although Baby experiences pleasure and pain (in the sense that tokens

appear in affect streams) it does not currently act to achieve pleasure or avoid pain. In fact, Baby's actions are either random—swinging its arms, turning its head—or probabilistic reactions to tokens in its streams; for example, it will probably cry if the HUNGER stream contains MODERATELY-HUNGRY. Baby has no goals and thus no basis for focusing attention on streams that are relevant to its aims. Although Baby might learn associations between hunger, crying, Mommy and comfort, it is constitutionally incapable of recognizing comfort as a goal or planning to cry to bring Mommy and achieve comfort. It would be simple to impose a planner on Baby. We could give it means-ends analysis, and it could attempt to reduce the difference between the current state (moderately hungry) and a goal state (comfort) by taking some action. But this presumes Baby knows what an action is and does, and we are very unwilling to impose this ontological primitive on Baby if there is any chance Baby might learn it by interacting with its environment. Similarly, we don't want to impose a planner on Baby if Baby might learn associations that, when triggered, activated and attended, give rise to actions that avoid pain and achieve pleasure. In sum, Baby lacks the most rudimentary notions of goal and action, and it remains to be seen whether they can be learned. Until then, we cannot rely on goals to focus attention.

Some other attentional mechanisms are closer at hand. We are currently implementing habituation and a "do it again" heuristic. Habituation means Baby is less likely to attend to an attended fluent as time goes on. The "do it again" heuristic says if an attended fluent includes a positive affect stream and a somatic stream in its scope, then repeat the action in the somatic stream. For example, a fluent might include streams for grasping, waving, the sight and sound of a rattle, and interest (which is a positive affect). As long as Baby waves the rattle and it makes noise, this fluent will be active. Because it is associated with positive affect, Baby will repeat the somatic component of the fluent (waving). Eventually, Baby will cease to attend to this fluent because of habituation, but in the meantime, it will wave the rattle a lot. The parallels to Piagetian circular reactions are intriguing.

Because the "do it again" heuristic keeps a fluent attended for a relatively long time (relative to Baby's current, random, activities, that is), it provides the opportunity to learn associations between the attended fluent and others. It also helps Baby to learn associations with events that would otherwise be very rare. This is how Piaget's notion of accommodation might emerge: an attended fluent is associated with others, forming larger fluents. Because the attended fluent persists (thanks to the "do it again" heuristic), Baby is apt to learn new fluents that are elaborations of it, rather than something completely unrelated.

7 Conclusion

Let us step back from the speculations of the previous section to assess what we have actually accomplished. The MSDD algorithm learns predictive relationships among multitokens that comprise streams or features, for the purpose of categorization and anticipating successor multitokens. Our fluent-learning algorithm learns simple fluents and four kinds of temporal associations between them. We have generalized the MSDD algorithm, which learns a generalization hierarchy of associations between feature vectors. The fluent-learning algorithm learns a generalization hierarchy of associations between fluents, so for the first time, we can learn relationships between processes, activities, experiences with temporal extent. We think these relationships provide the mental raw material for planning by imagination, as described in section 1. An optimistic view, then, is that Baby lacks only the notions of goals and actions, and given these, it will be able to plan by imagination. In this view, Baby's rules differ in complexity but not in kind from our knowledge that when we hoist a rickety kitchen cabinet, its doors can swing out an hit us on the nose.

In fact, we don't know whether relationships between fluents are different in kind from the knowledge we use to imagine activities. The structure of fluents (and relationships among them) seems minimal compared with the rich structures in, say, the case-based planning literature (e.g., Hammond, 1986). We do not yet know how Baby "carves up" its experiences. We tend to think of processes in terms of a beginning, middle, and end, and there's considerable consensus about which activities are which. We "just know" that hunger leads to crying, which brings Mommy and results in comfort. Baby might learn instead that eating leads to hunger, which we "just know is wrong," even though it is highly predictive regularity. It remains to be seen how Baby will carve up its world.

Baby is not a tabula rasa, of course, and its prior structure will influence what it learns. One source of bias is Baby's perceptual system, which places "interpretation" tokens in streams. For example, when a green rattle is within Baby's field of view, the to-

ken GREEN is placed in the SIGHT-COLOR stream and RATTLE-SHAPE goes in the SIGHT-SHAPE stream. Objects can share properties, such as greenness, and some non-rattles have a RATTLE-SHAPE, so perception is not without ambiguity, but the fact remains that Baby perceives the world according to a set of streams and token values that we designed. Another source of bias is Baby's attentional mechanism. We have already described how the "do it again" heuristic is expected to result in elaborating fluents, that is, learning new fluents by modifying old ones. Perhaps the most important source of bias is something Baby currently lacks: the ability to group fluents into classes based on sensory features. As we noted earlier, this ability is essential in Mandler's theory of conceptual development, but completely absent from Baby.

8 References

Computational Intelligence. 1994. The Imagery Debate Revisited. Special issue of *Computational Intelligence*, Vol. 9, No. 4.

Gibson, J .J. 1979. *The Ecological Approach to Visual Perception.* Boston: Houghton-Mifflin.

Hammond., K. 1986. CHEF: A Model of Case-based Planning. *Proceedings of the Fifth National Conference on Artificial Intelligence.* pp. 261-271.

A. E. Howe and P. R. Cohen. Understanding Planner Behavior. To appear in *AI Journal*, 1995.

J. M. Mandler. How to Build a Baby: II. Conceptual Primitives. *Psychological Review*, 1992, Vol. 99, No. 4, pp. 587-604.

J. McCarthy and P. J. Hayes. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In B. Meltzer and D. Michie, *Machine Intelligence IV.* Elsevier.

T. Oates, D. E. Gregory and P. R. Cohen. Detecting Complex Dependencies in Data. To appear in *Proceedings of the Fifth International Workshop on AI and Statistics*, 1995.

Z. Zheng. A benchmark for classifier learning. Technical Report from Basser Department of Computer Science, University of Sydney, NSW. 1994.

9 Acknowledgments

We thank two anonymous reviewers for their insightful and encouraging comments. The work on MSDD was supported by ARPA/RL Contract F30602-93-C-0100.