Benefits of Incorporating a Stream of Thought: A Case Study

Abhijit Mahabal (amahabal@cs.indiana.edu)
Department of Computer Science, 150 S. Woodlawn Avenue
Bloomington, IN 47401 USA

Abstract

I describe here a computer model of high-level perception named Seqsee that extends integer sequences in a human-like way. It is similar in architecture to Copycat (which solves letter analogy puzzles), but with a novel addition, an explicit stream of thought. Invoked by psychologists and philosophers alike, the stream has been used to explain limited capacity and even consciousness. The addition of the stream of thought to the Copycat architecture produces tangible benefits. I would argue here that the stream greatly improves similarity discovery and naturally facilitates self-watching.

Keywords: Stream of thought; Fringe; Sequence extension; Computer model

Introduction

The stream of thought is a central and highly cited part of William James' extensive writing. Baars (1988; 1996) and Mangan (1993) credit the serial procession of thoughts through the theater of the mind with many of our cognitive abilities, and Dan Dennett (1993) similarly gives the credit for consciousness to serial processing made possible by the *Joycean machine.*[1]

However, apart from the work of Eric Mueller (1990) on modeling daydreaming, and the computer program IDA[2] (Franklin, Kelemen, & McCauley, 1998), computational models of cognition have not explicitly implemented the stream.

This paper describes the computer program *Seqsee* for integer sequence perception. An input sequence may be "1, 1, 2, 1, 2, 3…" to which the program may respond with "Are the next terms 1, 2, 3, and 4?" and subsequently describes the sequence in words. Seqsee is able to extend several sequences including all those listed in Table 1. Seqsee is intended to solve sequences in a human like way. It uses a stream of thought, and the stream plays an important role in its discovery of the sequence structure.

In the initial stages of its design and implementation, Seqsee did not have a stream. As Seqsee was extended to handle more sequences, it became apparent that temporality plays an important role in sequence perception. Recency effects are also relevant in sequence understanding, and as the stream of thought could easily account for these, it was added to the mix. Having also worked on the stream-less version of Seqsee has been helpful in understanding the specific benefits that the stream confers.

Table 1: Examples of Sequences Solved by Seqsee

#	Sequence
(a)	1, 2, 3, 1, 2, 3, 4, 1, 2, 3, 4, 5, …
(b)	1, 7, 12, 2, 8, 13, 3, 9, 14, …
(c)	1, 7, 1, 2, 8, 1, 2, 3, 9, …
(d)	1, 1, 2, 3, 1, 2, 2, 3, 4, 1, 2, 3, 3, 4, 5, …
(e)	5, 6, 6, 7, 4, 5, 6, 7, 7, 8, 3, 4, 5, 6, 7, 8, 8, 9, …
(f)	1, 2, 3, 17, 4, 5, 6, 17, 17, 7, 8, 9, 17, 17, 17, …
(g)	1, 2, 3, 1, 2, 2, 3, 4, 1, 2, 3, 3, 3, 4, 5, …

Novel contributions of the work presented here include:

1. Theoretical contributions:
    a. A simple implementation of the stream, including the fringe (described later). This implementation can easily be applied in domains beyond sequence recognition.
    b. A novel method for similarity discovery, achieved by utilizing the stream, and by relying on temporal proximity.
    c. Implementation of self-watching mechanisms within the stream-based process.
2. A working program—Seqsee— that can extend integer sequences.

In the following sections, after briefly describing the sequences that Seqsee is intended to tackle, I describe the architecture of the stream-less Seqsee. I then put the stream back into the mix and demonstrate in successive sections how it is useful. This treatment allows me to compare the performance of Seqsee with and without the stream. Along the way, I will also describe how without the stream some issues become insoluble or at least substantially harder.

The Domain of Seqsee

The inputs to the program are the first few terms of an integer pattern sequence. The output consists of queries ("Are the next three terms 5, 6, and 7?") and statements ("I think I get it: the next few terms are 1, 1, 1, and 2"). Hofstadter (1995) describes this problem domain.

[1] A virtual machine that makes almost serial processing possible on parallel hardware of the brain, described by Dennett.

[2] IDA implements Baars' Global Workspace theory of consciousness to handle billet assignment for the U.S. Navy.

Some of the sequences in Table 1 (all except *b*), while still being easily understandable, are more complex than those handled by earlier sequence extension work like that of Simon and Kotovsky (1963). Their program only dealt with sequences of a fixed period (like sequence *b*).

The examples in Table 1 have been chosen to display the cognitive richness of the domain. I will focus on sequence *b* for a while. For a human, its 3-ness jumps out. It becomes clear very soon that the right thing to do is to see the sequence as "[1, 7, 12], [2, 8, 13], [3, 9, 14] …" What we certainly do not do is to try out various ways of splitting the sequence. We do not split the sequence, for instance, as "[1, 7], [12, 2], [8, 13]," just to see if it works. Reasons that we do not split it this way include (a) it is unmotivated, and trying out things without reason has a very slender chance of succeeding, (b) a huge number of possible splits exists, and trying these out by brute force is cognitively implausible, and (c) there are sequences which are indeed understandable by splitting into groups of two, but naïve attempts to do so would fail. For example, sequence *c* can be understood as "[1, 7], [[1, 2], 8], [[1, 2, 3], 9]…" but not naïvely as "[1, 7], [1, 2], [8, 1], [2, 3]…"

One problem facing us, then, is how that 3-ness can initially emerge in the absence of any explicit attempt to look for it.

Another instance of the richness of the domain is sequence *d* which was designed so that it could be seen as a special case of sequence *a*. In order for this to be correctly seen, Seqsee must be able to see something (for example, "1, 2, 2, 3, 4") as something else (a blemished "1, 2, 3, 4"). This ability of *seeing as* has been suggested to be crucial to intelligence (for example, Hofstadter, 1995).

The Architecture without the Stream

Stripped of its stream, Seqsee's architecture is identical to that of Copycat, though the problem domain is very different. Copycat is a program by Mitchell (1993) that solves letter string analogy puzzles. Its architecture is described in detail elsewhere (Hofstadter & FARG, 1995; Mitchell, 1993), and only the relevant aspects are described here.

The way Copycat finds its way to the solution is by noticing various structures within its input. If three consecutive letters in its input are *a, b,* and *c*, then it might notice the relation between *a* and *b*, namely that the latter is the successor of the former, and create a structure called a *bond* connecting the two. After discovering the relation between *b* and *c*, it may also discover the "*successor group" abc*. By being able to discover ever-larger chunks, it can finally make sense of the entire input. These structures—bonds, groups, and others—are stored in the *Workspace*. All work is done small pieces of code called *codelets*. The workspace is like a blackboard that all codelets can read from or write to.

Copycat consists of a *Slipnet* (the long-term memory), the workspace (where perceived structures are stored), and a *coderack* (where codelets await their turn). Associated with items in these components are numbers between 0 and 100 indicating desirability: workspace structures have strength and salience, Slipnet nodes have activation and depth, and the codelets have urgencies. All operations in Copycat are stochastic; codelets choose objects to operate on randomly, but biased by strength or salience; and codelets are themselves chosen from the coderack randomly, biased by urgencies. The central cognitive loop of Copycat looks thus:

i. Choose a codelet,
ii. Do its bidding,
iii. Repeat.

The bidding of a codelet may include creating various structures in the workspace (like *bonds* describing the relation between objects or *groups* that chunk together several objects). It may also include destroying some structures, or even adding more codelets to the coderack. Sometimes, if the objects it randomly selected are inappropriate for the codelet, the codelet just fizzles without having done anything.

In Copycat (and in Seqsee), there is no central executive guiding the program along: at any time, one of the codelets is chosen randomly and executed. Different codelets may pull the system in different directions, thereby allowing Copycat to explore several avenues at once. A codelet may add other codelets to the Coderack, and assign urgencies to the new addition. Such urgencies represent an estimate of the importance of the new codelets made by the existing codelet. Promising avenues (as estimated mechanically by codelets that have recently run) are explored more vigorously, this feat being accomplished by there being more codelets (or codelets with higher urgency) pulling in that direction.

The Stream in Seqsee

Thoughts

Seqsee contains all the architectural components of Copycat, including the workspace, the Slipnet and the coderack. Besides codelets, Seqsee features *thoughts*. Thoughts are a special case of codelets. While the function of a typical codelet is very tiny and usually myopic in the sense that it is concerned with only a small part of the workspace, a typical thought can have a more global impact. An example would clarify the difference. In the sequence *b*, it may happen that the program has been able to form chunks in the following way: '1, [7, 12, 2], [8, 13, 3] …' One possible thought in this situation is "I have started on the wrong foot." This is a global realization in that it refers to the entire sequence. Once the program realizes that it has started on the wrong foot, it knows of things it can try to fix the problem.

Other examples of thoughts—suitably transcribed into English—include "Am I in a rut?" and "Am I certain of this solution?" Some thoughts correspond to objects in the workspace; these thoughts are Seqsee's way of focusing on the object, and their English transcription is "what can I do with this object?"

By contrast, most codelets do a small task quickly, and are more local. Clearly, thoughts and (other) codelets lie along a spectrum of complexity and do not form a strict dichotomy. The difference is pointed out here because some types of codelets have been classified (by me) as thoughts, and Seqsee has mechanisms to automatically be sensitive to repetition in thoughts.

Various analogies may be made between the thought/codelet spectrum and ideas in the writings of James and Baars. Codelets are like the unconscious specialized processors in Baars' Global Workspace theory. Thoughts are more like the processors that march onto the stage.

Intuitively, thoughts are somewhat like the resting places in William James' analogy of the stream of thought with the perching and flight of a bird. The resting places "can be held before the mind for an indefinite time, and contemplated without changing; the places of flight are filled with thoughts of relations, static or dynamic, that for the most part obtain between the matters contemplated in the periods of comparative rest." In Seqsee, recent thoughts are remembered in the stream, whereas codelets that are not thoughts just do their job and disappear.

The Fringe

William James (1890) often mentioned a "penumbra" or "fringe" of "vague" experiences, sometimes using the terms "psychic overtones" or "suffusion". I use the word fringe in a way consistent with William James' use of the term. The fringe of a thought can also be seen as a set of concepts that a given thought may evoke.

The fringe will play a central role in the story being told. The concept of the fringe has seen an upsurge of interest in recent years. Bruce Mangan (1993) describes several uses of the fringe, including one that concerns us here the most: *it works to radically condense context information*.

Each thought in Seqsee—whether or not it corresponds to some workspace object—has a fringe of related content. For example, if the group '1 2 3' has been seen as an ascending group, the fringe of the corresponding thought includes the concept *ascending group*.

Assume that Seqsee has been working on sequence *b* for a short while and its most recent thought was about the second element ("7"). In the near future, what effect should this have on the subsequent behavior of Seqsee?

The recent thought "7" is part of the context in which subsequent events are interpreted. Seeing another 7, for instance, may alert Seqsee to the possibility of a significant long-distance relationship between parts of the sequence. If the sequence had been "3, 7, 3, 7, 3, 7…" then noticing the repetition of the 7s would be one step toward the 2-ness standing out.

However, it is not just another 7 that should get Seqsee to perk its ears: anything "*like 7*" should also work. It is not clear at the outset just what "*like 7*" means; it is partly dependent on the sequence. What is "*like 7*" in the following two sequences is quite different:

- in sequence *b,* and
- in "3, 7, 3, 3, 7, 7, 3, 3, 3, 7, 7, 7…"

It will turn out that the fringe may serve not just to discover similarity but partly also to create it.

The fringes of recent thoughts form the context in Seqsee. I use the term *context* to mean, following Baars (1988), "the inner world that shape experiences and voluntary actions." Contexts are a key idea in Baars' cognitive theory of consciousness.

For Seqsee, the fringe of a 7 in the sequence contains the numbers 7, and to a lesser extent, 6 and 8. Seqsee does not deal with such properties as *primeness*, but if it did, the fringe of some 7 in the sequence may also contain "is prime," *provided that that 7 has been so seen*. The proviso "provided it has been so seen" is important. Not every time we see a 7 is its primality impressed upon us. In the sequence "1, 2, 3, 4…" the primality of the 7 is immaterial, and unless we have been thinking of primes recently we will not consciously think of its primality.

Note that the thought was about a specific 7 in the sequence, and not about the number 7: two different 7s in the same sequence can play distinct roles and have different fringes, as in the sequence "4, 7, 5, 7, 6, 7, 7, 7…," thoughts corresponding to each of the five 7s in the sequence would have a different fringe, and that of the highlighted 7 would be the most different.

The fringe of the 7 also contains the position in the sequence and (to a lesser extent) the positions around it. All this means that, in the near future, anything that is near the 7 or is something like a 7 will be treated slightly differently, by a mechanism to be described in the next section.

The Action-Fringe

There is another way in which the current thought has an effect on the flow of the program. The closest available word that I can find to describe this effect is *affordances*, though I hesitate to make this non-traditional use of the overloaded term. Designers of artifacts choose the interface with care, to make it obvious what actions can be taken with each component: this knob can be turned, that blue underlined text can be clicked, and so forth. A similar story also holds for things inside our minds. Some thoughts trigger others in specific ways born of habit. For example, the realization while solving a math problem that mathematical induction might work is usually followed (for me, at least) by the thought about what the base case would be. It is not always so clear-cut, of course. The realization of the potential solution via induction does not necessarily lead to base case search: instead, it sometimes leads to thinking about whether the induction step is feasible, or even to the reevaluation of the appropriateness of induction. It is almost as if a plethora of choices for potential action is available to us. We could think of this as "the fringe of potential actions" surrounding a thought.

Thoughts about an object also have such an action fringe. For example, when somebody gives me a puzzle whose statement contains a large number (say, 520) I sometimes

find myself factorizing it. The number, in that context, affords factorization.

For each thought in Seqsee, the action fringe is calculated. For a thought corresponding to a group in the workspace, this includes attempting to extend it. In sequence *a*, for instance, a group of the last three elements shown ("3, 4, 5") may be formed, and when it is the current thought the program may add codelets to try to extend it either left or right. Successful extension to the left would result in seeing all of "1, 2, 3, 4, 5" as a group. Attempting to extend it to the right may result in Seqsee incorrectly asking[3] if the next term is "6." For the thought "I am in a rut," the action fringe includes codelets that destroy weak structures[4] in the workspace.

To extend Seqsee to incorporate learning, habituation by means of learning action fringes based on what has "worked" in the past will be required. This will involve being able to assign credit or blame to thoughts in the recent past, and is a hard problem.

Interaction between Thoughts and Codelets

The coderack in Seqsee has been modified to contain, apart from the codelets, up to one thought. At any given time step, the thought or some codelet may be chosen to be the next to run. Codelets, when run, may add more codelets to the coderack, or add a thought. When a thought becomes the current thought, it can cause the addition of more codelets or a thought to the coderack. Codelets and thoughts are thus completely intertwined. The central cognitive loop of Seqsee is:

i. Choose a codelet or a thought from the coderack.
ii. If codelet, do its bidding.
iii. If thought:
    a. Calculate the fringe and the action fringe.
    b. Check for fringe-overlap with recent thoughts, and add to the set of potential actions accordingly[5].
    c. Launch some codelets or schedule a thought from amongst the actions.
iv. Repeat

In step (i) above, a parameter controls the likelihood of a codelet being chosen (as opposed to a thought being chosen). If this parameter is set at 100%, it has the effect of making the stream effectively nonexistent, and we are left with the same cognitive loop as that of Copycat. Changing this parameter has the effect of controlling the level of serial versus parallel processing. I have not yet experimented with various settings of this parameter, and have left it fixed at 70% so far. The possibility of codelets within Seqsee dynamically adjusting this has also not been explored.

Dennett argues that there cannot be a single stream of thought, however wide. Instead, he argues, there are multiple channels in which the specialist circuits try various things in parallel pandemonium, creating multiple drafts. This description is not inconsistent with Seqsee; the n^{th} thought does not fully determine the $n+1^{st}$ thought, and between any two thoughts, there can be an entire "pandemonium" of codelet activity.

The Discovery of Similarity

When a new thought arrives, its fringe is calculated. If this overlaps enough with one of the recent thoughts' fringes, this is taken to be a cue to explore the relatedness of the two thoughts, and codelets are launched to explore the similarity. If the overlap is with more than one recent thought, Seqsee chooses one of those biased by the amount of overlap.

While calculating fringe overlap, the overlapping items are not weighed equally: Seqsee is biased toward abstract similarities, and if two objects are both *blemished ascending* (for example, "2, 3, 4, 4, 5"), their similarity is judged more compelling than the similarity between two objects that are both of length 5.[6]

What is important in a given problem-solving situation is not known at the outset. As more structure is uncovered, the importance of every concept will wax and wane. Correspondingly, the fringe of a new thought reflects the current biases. If ascending groups seem to be in vogue, then two ascending groups would have a greater fringe overlap.

Copycat does not have a stream; it is instructive to compare how similarity-discovery proceeds there. Copycat contains codelets that look for similarity. One example is the *bond-scout*. This codelet chooses an object at random, then chooses a neighbor of this object, and checks if the two are similar. If they are, it launches other codelets that will actually create a bond. If they are not, however, the codelet just fizzles out. The more codelets that fizzle out, the more wasted motion there is.

The stream eliminates the need for bond scouts. The role of the two objects chosen at random is played by two thoughts that enter the stream in close temporal proximity. This is good because (a) Seqsee pays attention to the pair only if their fringes overlap (thereby strongly suggesting relatedness), and there is less chance of wasted motion, (b) less wasted motion allows a deeper exploration in the same amount of time, and more initially-less-promising avenues can be explored, (c) the two thoughts need not be about physically proximate objects, and long-distance relations can be perceived. This is harder without the stream, because two randomly chosen distant objects are even less likely to

[3] The probability of that being asked is low, but non-zero.

[4] And sometimes, but with much lesser probability, strong structures.

[5] What extra actions are added in case of overlap is described in detail in sections "*The discovery of similarity*" and "*Self-watching*."

[6] Provided, of course, that the activation of *length* is not much higher than the activation of *blemished ascending*.

be related, and finally, (d) the two thoughts need not even be about workspace objects! The stream facilitates the discovery of far general similarity, as explained in the next paragraph. It is unclear how this can be done without something like the stream.

Sometimes we have a problem in our minds that we are not actively thinking about, but we immediately recognize the solution if it presents itself before us, as might have happened with Archimedes who was faced with the problem of finding the volume of his king's crown. In the bath, he noted that a body displaced water, probably equal to its volume. The "problem" and the "solution" both related to the volume, and hence were similar. In Seqsee-speak, the fringe of both contained volume and would have led Seqsee to connect the two.

If this seems too vague, consider the sequence named *marching doubler starting midstride*, "1, 2, 3, 3, 1, 1, 2, 3, 1, 2, 2, 3, 1, 2, 3, 3, 1, 1, 2, 3…," and imagine that Seqsee had formed a group of the middle 12 elements [[[1, 1], 2, 3], [1, [2, 2], 3], [1, 2, [3, 3]]]. Extending this group left is difficult, as it involves falling over the left edge. But whatever the leftward extension is, it must be a blemished 123. Finding such a blemished 123 immediately to the left of the big group is the "problem," and the fringe of any group that is read in and fits the description would greatly overlap with the problem, and thus be seen as a solution. Thus, the problem works like an "*active gap*[7]," and provides a context which colors subsequent objects that are seen

How Sequence *b* is Solved

Let us recall that sequence *b* is "1, 7, 12, 2, 8, 13, 3, 9, 14 …" As Seqsee is stochastic, the blow-by-blow analysis of different runs on the same problem will not be identical; what is presented is one likely scenario.

Seqsee has a codelet called *Reader* that reads objects from the workspace. An object just to the right of the last object read[8] is chosen with high probability, but any object has some chance of being selected. Thus, a procession of objects files through the stream.

If it happens that the second element ("7") is read after the fifth element ("8") had recently been read, because of the fringe overlap of the two thoughts a codelet is launched that checks if a bond can be formed between the two. This succeeds, as 8 is a successor of 7. This newly created bond can also now be read into the stream by the Reader codelet.

The action fringe of a bond that is long-distance (meaning that, as in the bond above, the two objects are not adjacent) includes finding the number of intermediate objects between the two ends. The Slipnet node corresponding to this distance is lightly activated.

As several other pairs are seen with the same distance, the activation of this node[9] keeps going up. When sufficiently high, when a long-distance bond is the current thought, a group can be formed consisting of objects from one end to just before the other end. Thus, if many bonds with two intervening objects are seen, the 7→8 bond can result in the formation of the group "[7, 12, 2]." The fringe of such a group would overlap with any other group consisting of three elements formed in this manner, and higher-level structures can be discovered. This may lead to understanding the sequence as "1, [7, 12, 2], [8, 13, 3]…" which is enough to extend the sequence correctly. People are easily able to "shift the boundaries" to get the "correct" parsing, and how Seqsee re-parses the sequence is described in the next section. About a third of the time, Seqsee's initial parsing is already the correct one.

In sequence *c*, "1, 7, 12, 1, 2, 8, 13, 1, 2, 3, 9, 14…" this mechanism leads to light activation of the node corresponding to distance five, because of the 13→14 bond. However, after the "1, 2, 3" has been seen as a group, the distance between the 13 and the 14 is three. Even though Seqsee feels a 5-ness, it is not a strong feeling as very few distance-5 bonds are seen. In a sequence like "2, 7, 2, 7, 2, 7…" concepts corresponding to relations with ends distances 2, 4 or 6 apart get activated, and Seqsee sometimes sees this sequence as an endless procession of [2, 7, 2, 7].

Serendipity

The way similarity between two things is discovered in Seqsee seems to be via happy coincidences that brought the two objects into the stream at nearby times. Is such reliance on happy coincidences justified?

Once a few relations are discovered, Seqsee can find other relations by extending in either direction what it has found, using action fringes. This leaves me with needing to explain how at least a few relations are guaranteed to be seen by the similarity discovery mechanism.

A run may go on for a few thousand codelets, and the likelihood of any two objects to be read within a few codelets of each other is high. Moreover, as only a few of these coincidences must occur for Seqsee to understand the sequence, the probability of enough relations being discovered is high.

Seqsee does not understand all sequences easily. It struggles, for example, with sequence *d*, successfully extending it roughly half the time within a running time of 10,000 codelets. However, this poor performance is not caused by a lack of coexistence of related pieces within the stream; the intrinsic hardness of the sequence is a more likely cause.

Another justification for reliance on serendipity is the role it seems to play in the way we function. The book

[7] This phrase is used by William James to describe the phenomenology when we are trying to recall a forgotten name, and "A sort of wraith of the name is in it, beckoning us in a given direction. […] The rhythm of a lost word may be there without a sound to clothe it"

[8] There can be more than one: the very next number in the sequence, or any of the groups of different sizes starting there.

[9] The node "*distance-5 bonds*" is active when several bonds with ends separated by 5 objects have been seen.

"serendipity: accidental discoveries in science" (Roberts, 1989) gives hundreds of examples of how two previously unrelated elements when luckily seen together have led to significant scientific advances. In the case of Goodyear, for instance, the unrelated elements rubber and heat were accidentally brought together when the piece of rubber he was working on touched a hot stove, leading to the discovery of vulcanization. The prepared mind of Goodyear clearly played a role in the discovery, but the number of possible things that might actually be connected is so vast that nobody can make a systematic exploration of all possibilities, and many discoveries must come without explicitly being sought.

This is equally true about Seqsee. Especially in sequences like *c* and *d* where several of the important bonds are long-distance; their discovery cannot be achieved by brute force. Two random objects arbitrarily far from each other should not be tested for relatedness; such a testing is a wastage of resources as the likelihood of success is slim. In fact, Copycat does not look for long distance bonds and consequently cannot solve such problems. The stream of thought helps Seqsee by allowing it to be sensitive to relatedness of temporally nearby thoughts and raise the likelihood that any two things tested for relatedness actually are related. Thus, testing long distance relationships is no longer a risky proposition, as only relations "pre-screened" by the stream are tested.

Self-Watching

In some sequences, Seqsee (and humans) can be lead down a garden path. An example from Hofstadter (1995) that illustrates this is "2, 1, 2, 2, 2, 2, 2, 3, 2…" The group of five 2s forms quickly, but it is the wrong group and leads nowhere. During the course of the run, it is destroyed but is soon re-created by Seqsee. If this cycle continues, Seqsee would never be able to solve this sequence. People, too, are caught in this loop but noticing themselves going in circles, they break out of the loop effortlessly.

Marshall (and Hofstadter before him) has eloquently described the importance of self-watching in intelligent behavior (Marshall, 2002). Marshall's method of achieving self-watching used something like the stream for this task, and he even uses the phrase *train of thought* to describe the data structure used. This data structure ("the *temporal trace")* is like a history of important events that have happened during the run. A few codelets keep scanning the trace for repetitions, and if found, launch other codelets that make it less likely for the offending repetition to recur.

Seqsee's stream is *active*, and the mechanisms of repetition detection are in-built, but the underlying motivation is the same. The stream subsumes the role of the trace, with no extra effort. In an earlier section, I described how similarity discovery between thoughts works. Fringe overlap was described as a good thing. After all, it allowed Seqsee to uncover the underlying logic of the sequence. However, fringe overlap can also be bad: if Seqsee is having the exact same thought repeatedly, the fringe is bound to overlap. If such repetition happens, extra actions to handle the situation would be suggested in step iii (b) of the cognitive loop.

I do not suggest any extra mechanisms beyond those suggested by Marshall to decrease going in circles, but note that realizing the circularity in its own actions is streamlined in Seqsee, and utilizes no extra structures.

Moreover, approximate matching is easy for the stream, and hence the re-occurrence of almost the same thought can be detected.

Acknowledgments

This research is supported by funding from the Center for Research in Concepts and Cognition, Indiana University. I would like to thank Douglas Hofstadter and Chris Honey for useful discussions.

References

Baars, B. J. (1988). *A Cognitive Theory of Consciousness*: Cambridge University Press.

Baars, B. J. (1996). *In the Theater of Consciousness: The Workspace of the Mind*: Oxford Univ Pr on Demand.

Dennett, D. C. (1993). *Consciousness explained*: Penguin Books.

Franklin, S., Kelemen, A., & McCauley, L. (1998). IDA: a cognitive agent architecture. *Systems, Man, and Cybernetics, 1998. 1998 IEEE International Conference on, 3*.

Hofstadter, D. R. (1995). On seeing A's and seeing As. *Stanford Humanities Review, 4*(2), 109-121.

Hofstadter, D. R., & FARG. (1995). *Fluid concepts & creative analogies : computer models of the fundamental mechanisms of thought*. New York: Basic Books.

James, W. (1890). *The Principles of Psychology*: Dover Publications.

Mangan, B. (1993). Taking phenomenology seriously: The "fringe" and its implications for cognitive research *Consciousness and cognition 2*(2).

Marshall, J. B. (2002). Metacat: A Self-Watching Cognitive Architecture for Analogy-Making. *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, 631–636.

Mitchell, M. (1993). *Analogy Making as Perception*. Cambridge, MA: Bradford Books/MIT Press.

Mueller, E. T. (1990). *Daydreaming in Humans and Machines: A Computer Model of the Stream of Thought*: Ablex Pub. Corp.

Roberts, R. M. (1989). *Serendipity: Accidental Discoveries in Science*: Wiley.

Simon, H. A., & Kotovsky, K. (1963). Human Acquisition of Concepts for Sequential Patterns. *Psychological Review, 70*(6), 534-546.