

Seqsee: Yet Another Cognitive Architecture

(Ph.D. Proposal)

Abhijit Mahabal

Oct 6, 2006

Introduction

My Ph.D. project may be looked at from two related but quite distinct perspectives. The first view, which I call version-A, could seem to a distant observer to be all that my project is about:

Version A. I am designing and implementing a computer system that extends pattern sequences in human-like ways.

The second view—version-B—is closer to my main interest in the project:

Version B. Simulating how we think and all the spell-it-out that such an endeavor entails is likely to bring to the fore problems and issues that can otherwise be overlooked, and writing the program, thus, is merely a cover for exploring deeper.

There is also a third version that is important to me that I wish not to tie to my Ph.D.:

Version C. I am designing and implementing a system to solve some hard problems not easily amenable to traditional computer science, by reverse engineering, as it were, some aspects of the human mind. Seqsee is but a test case.

My reasons for not making this a requirement is that in order to claim any generality of this engineering approach I would need to show that it works on other things besides sequences. Modifying the program to work for other cases would (hopefully) not be tough theoretically, but it would still be a lot of work.

Contents

1	The Context of Seqsee	2
2	The Task in Greater Detail	4
2.1	The Domain	4
2.2	A Wish-List of Sequences	4
2.3	More than Just Extending	6
3	How Seqsee Currently Works	6
3.1	The Workspace and the Codelets	6
3.2	The Discovery of Similarity	7
3.2.1	The Fringe	7
3.2.2	Tagging	8
3.2.3	Categories	9
3.3	Affordances	9
3.4	The Stream	10
4	A Tentative Time-Frame	11

1 The Context of Seqsee

This section describes other work that is similar either in architecture, in goals or in idea-space. This work is being carried out in the Fluid Analogies Research Group (FARG), and is highly influenced by Hofstadter and the other FARGonauts. In methodology, in architecture, and in the general types of issues tackled it is strikingly similar to Copycat and other FARG projects. It draws most substantially from Mitchell (1990), Marshall (1999) and Hofstadter (1995). Chapter 1 of this last cited book describes the project of sequence extension (as envisioned by Hofstadter) and the evolution of the project's goals.

A larger body of work within which the various FARG implementations lie is that of programs inspired by the Blackboard architecture of the HEARSAY-II system (Reddy, Erman, Fennel, and Neely 1973), which in turn can be said to belong to systems with a pandemonium style architecture (Dennett 1991), characterized by large tasks being achieved by large numbers of small relatively independent not-very-smart agents.

Seqsee is by no means the first program to attempt this task. There have been several other projects that have shared the version-A goals of Seqsee. Most notably, Simon and Kotovsky (1963) worked on such a program. Though the claims of their paper are big (their italics: “*normal adult beings have stored in memory a program capable of interpreting and executing descriptions of serial patterns. In its essential structure, the program is like the one we have just described*”), the program does not seem to do much. I have on my website (Mahabal 2006) my 30-line program that solves all the 25 problems which the Simon-Kotovsky system was tested on (where as their program solved 22). While my 30-line version can not be called *cognitively plausible*, I think that neither can theirs. Their system deals only with periodic patterns, and even ‘1 1 2 1 2 3 1 2 3 4 . . .’ is considered out of bounds.

Marsha Meredith’s Seek-Whence (Meredith 1986) was another sequence program. If I have my history right, it was difficulties encountered in the sequence domain that were the genesis of the Copycat family of programs. Seqsee has surpassed Seek-Whence in the sequences that it can handle.

As Seqsee is intended to extend sequences in a *human-like* way, its construction has been influenced by several cognitive scientists’ ideas, and the following list is presented as Seqsee’s context in idea-space:

- The program gets its traction from tagging various intermediate objects with labels. These labels help Seqsee notice high-level similarity between constructs. Several people including Dennett (1994) and Andy Clark (Clark 2001; Clark 1998) have described this role of “words” in human language, and how it aids cognition. The specific labels that I use are fluid categories of the sort described by Hofstadter. I will have much more to say about tagging in section 3.2.2.
- Noticing similarity between constructs also happens when the *fringe* of a construct overlaps another. James (1890) talks at length about fringes, and I describe how Seqsee uses them in section 3.2.1.
- The way thought leads to thought happens in Seqsee is via something not unlike Gibson’s affordances. This, and the somewhat related Roger Schank’s theory of scripts, is elaborated on in section 3.3.
- And finally, Seqsee features an explicit *stream of thought* (James 1890) and this—with hedges, qualifications and warnings about only loosely appropriate metaphor—is the subject of section 3.4.

2 The Task in Greater Detail

2.1 The Domain

Hofstadter's interest in sequence-extending programs goes back at least a quarter of a century, and the tasks intended for "the program" have shifted over time. In the original pre-Seek-Whence guise, a typical input for the program could have been, for instance, the sequence of alternate primes '2 5 11 17 23 ...'

Gradually Hofstadter's interest shifted away from such mathematical-knowledge intensive sequences to a smaller domain: that of sequences which involve no knowledge of arithmetics beyond the relationship of successor and predecessor and the ability to count. This meant outlawing such sequences as the primes, and even the addition-involving Fibonacci sequence. The shift to this stark domain is described by Hofstadter (1995, pages 48-49).

This restricted domain is by no means poor in sequences that tax the intellect, or even in sequences excellent for exploring program-cognition. For sequence k on page 5 in the next section, for example, it is interesting to contemplate how a program might navigate the 2 2 2 2 2 conundrum. Several other sequences require for their honest human-like understanding the ability to chunk (almost all examples in the next section) and even to see an object *as* another (examples $e, f, g,$ and h).

I shall not here try to delineate further the problems that could be the inputs for Seqsee, but leave that task instead to the dozen or so examples in the next section.

2.2 A Wish-List of Sequences

The following annotated listing of sequences is offered in order to add accountability: These are sequences that Seqsee should "get" before we can pronounce it to be *good enough*. The list starts out rather simply, but builds up tempo as it goes along.

Most of these sequences come from chapter 1 of Hofstadter (1995).

a. **1 2 3 4 ...**

Seqsee must get this, of course, but it should also get this rapidly even if a hundred terms are initially presented. Its performance should be no worse than linear. If it takes time quadratic (or even just super-linear) in the number of input terms, there is something seriously wrong with the model.

b. **1 1 2 2 3 3 ...**

c. **1 2 2 3 3 3 . . .**

Seqsee should see the initial “group of size 1”.

d. **12 13 14 11 12 13 14 10 11 12 13 14 . . .**

The program—and it is not aware of negative numbers—should know that it is going to hit a wall in the future.

e. **1 1 2 3 1 2 2 3 4 1 2 3 3 4 5 . . .**

I want Seqsee to be *able* to see the initial 1123 as a variant of 123, and likewise the following groups as variants of 1234 and 12345.

f. **1 1 2 3 1 2 2 3 1 2 3 3 1 1 2 3 1 2 2 3 1 2 3 3 . . .**

g. **1 1 2 3 1 2 2 3 1 2 3 3 1 2 2 3 1 1 2 3 1 2 2 3 . . .**

h. **1 1 2 3 1 2 2 3 1 2 3 3 1 2 3 1 2 3 1 2 3 . . .**

These three sequences are all “doubler” sequences: The first two were called marching- and bouncing-doubler, respectively, by Hofstadter. I call *h* the piercing-doubler.

As in the sequence *e* above, Seqsee should be able to see an endless list of 123 variants, with funny things happening in all three cases. As the names of these sequences indicate, people can perceive “motion” happening in these, and I’d like the program to be able to, too. It should also notice that all three are similar sequences, cut from the same cloth.

i. **2 10 1 3 11 2 4 12 3 . . .**

j. **2 10 2 3 11 2 3 4 12 . . .**

These interlaced sequences cannot be perceived merely by a rudimentary “try every n^{th} term and see if they fit together”, as can be seen from sequence *j*.

k. **2 1 2 2 2 2 2 3 2 2 4 2 . . .**

A confounding cousin of *i* and *j* above.

l. **1 0 0 1 1 1 0 0 0 0 . . .**

m. **1 2 3 1 2 2 3 1 2 2 2 3 . . .**

n. **5 4 3 4 5 4 3 4 5 . . .**

o. **1 2 3 3 4 4 5 5 5 6 6 6 . . .**

Somehow I feel that the last one above is going to require the greatest amount of work.

2.3 More than Just Extending

The goals of the project also include things apart from extending. Making variations on a theme is an example of such a goal. To take a simple example, a variation of the sequence ‘1 2 3 1 2 3 1 2 3 ...’ is the sequence ‘1 2 3 4 1 2 3 4 1 2 3 4 ...’. A more subtle example would be the variation of sequence f (the marching doubler) to produce the marching singler (‘1 2 2 3 3 1 1 2 3 3 1 1 2 2 3 ...’). Hofstadter (1985, chapter 12) discusses the general theme of variation-making.

It is important to note that how we perceive the sequence influences the variations we can make. The marching doubler can be perceived in a life-less way as the following 12 terms repeating indefinitely: 1 1 2 3 1 2 2 3 1 2 3 3. Seen this way without noticing the internal structure, one possible “faithful” variant would be the following 12 terms repeated indefinitely: 17 2 11 18 5 1 6 9 7 12 5 13. In some sense that would completely miss the point. Thus the issue of variation making is not divorced from the issue of sequence perception.

Another aim is that the program be reminded of earlier sequences that were similar. This functionality is similar to the *episodic memory* in Metacat (Marshall 1999).

A third aspect of going beyond just extending concerns meta-thoughts. Consider the question of when to stop. Whether we have found the answer or not is a judgment call, as is the question of whether we should give up. A central question that Metacat addressed beyond Copycat was that of self-watching, and self-watching will also be very important for Seqsee. It already has some meta-thoughts. Now that the other things are falling into place I can pay self-watching and meta-thoughts the attention that is due.

Most tasks in this section belong to the category *still to be done*, but they are absolutely crucial to the project, and would get a large chunk of my efforts in the coming days.

3 How Seqsee Currently Works

What follows is a broad brush stroke description of the working of Seqsee and its theoretical underpinnings. For want of space it skimps on details but I would be happy to elaborate on any of this during the actual presentation.

3.1 The Workspace and the Codelets

Seqsee, like Copycat and other FARG projects, has a workspace which is the arena where “perceived objects” live. These perceived objects include chunks that Seqsee has created out of the

input elements, the relations between various objects and other things besides. The workspace is almost exactly the same as its namesake in Copycat and I shall not comment on it further.

Codelets, too, are similar to their Copycat counterpart. Each codelet contains a small piece of code that does some microscopic act of cognition. Examples include checking if two objects are related and checking if an object belongs to a certain category.

The only different thing in Seqsee is a variety of codelets that can be run immediately rather than being stored on the coderack. This can of course be simulated by the Copycat codelet system by using codelets of extremely high urgency (thereby causing that codelet to get run almost immediately).

3.2 The Discovery of Similarity

The codelet *find-if-related* can look at two objects and determine—given its current knowledge of the two objects—if it can find a relation between the two. This section is *not* its story. Rather, it is the story of why some other part of Seqsee created this codelet to work on those two specific objects. In other words, speaking anthropomorphically, why somebody thought that these two objects had some chance of being related.

Here Seqsee diverges from Copycat. Copycat also has a codelet with a similar function: the *bond-scout*. The difference is that the *bond-scout* chooses its objects randomly (though preferring salient objects). Even the top-down cousin of the *bond-scout* chooses its objects randomly. *find-if-related*, on the other hand, gets called only on objects that *could* be similar. This has a major advantage in terms of how smoothly the discovery of structures proceeds. (Several other things in Seqsee are bottom-up, of course).

Before I describe how this sense of similarity comes about, a short digression. Note that each object in the workspace is also a concept. If you see a few numbers written on some white-board, then you may point to the first 7 on the board. That 7 is a concept, different from another 7 on the same white-board. These are temporary concepts in that they would almost certainly not enter the long-term memory, and similarly objects in the workspace are also quiet transient, but are real for that short duration. Now back to the *discovery-of-similarity* story, which unravels in three very related segments.

3.2.1 The Fringe

The notion that concepts have a fringe of floating associations is not new, going back to at least William James (James 1890)

Let us use the words *psychic overtones*, *suffusion*, or *fringe*, to designate the influence of a faint brain-process upon our thought, as it makes it aware of relations and objects but dimly perceived.

Several people have commented on the fringe, including Hofstadter (1997) in Chapter 10 *On words and their magical halos*. A particular example from this chapter is that of the concept *cheese*, and how it brings up images of either “oranged processed squares that come pre-sliced and pre-wrapped in plastic” or “a tray full of unprocessed cheeses, perhaps a hunk of Brie, some Gouda, some chèvre” depending on the context.

Concepts in Seqsee have fringes, and these are also context dependent. In Seqsee, a concept activates other concepts to various degrees and thus other concepts are in the fringe to different degrees. As objects in the workspace are concepts, they too have fringes.

I will hold on for the moment to what fringes contain, but in short two concepts are considered worth the attention of a *find-if-related* codelet if their fringes overlap. This can be made concrete with the following example. In the sequence ‘1 7 2 8 . . .’ the fringe of the object 7 contains the number seven and to a lesser extent the numbers eight and six. The fringe of the object 8 also contains the number eight and thus the two objects stand a chance of being perceived as related.

3.2.2 Tagging

The story for 7 and 8 above was simple. But how are the groups *123* and *1234* seen to be potentially similar? The answer is that when they have both been perceived as *ascending groups*—and not before—they appear similar to Seqsee. These tags are part of the fringe, and clearly, if both objects share a tag, their fringes overlap. The overlap is a matter of degrees and hence so is this feeling of similarity.

I think there is something cognitively interesting going on here that I would like to draw your attention to via a quote from Andy Clark (Clark 2001).

Learning a set of tags and labels is rather closely akin to acquiring a new perceptual modality. For like a perceptual modality, it renders certain features of our world concrete and salient, and allows us to target our thoughts (and learning algorithms) on a new domain of basic objects. This new domain compresses what were previously complex and unruly sensory patterns into simple objects. The simple objects can then be attended to in ways that quickly reveal further (otherwise hidden) patterns, as in the case of relations between relations.

In the case when the overlap of the fringes includes such tags there is more information to discover the relationship. Seqsee does not deal with prime numbers, but pretend for a moment that it does and that it has been presented with the sequence ‘2 3 5 7 11 ...’ If both 5 and 7 have been tagged as primes then in the process of relation finding the program may discover that 7 is the next prime after 5, and use this information to understand the sequence.

3.2.3 Categories

The third part of the story concerns what Seqsee uses as labels. These are *categories*, examples being *ascending group*, *lengthening relation* and *123*. What follows is a really short description of categories and the heavy lifting that they do in Seqsee.

Seqsee categories are not black-and-white. The category *123* will consider a *1 2 3* group in the workspace as a very good instance, and a *1 1 2 3* or even a *1 1 2 2 3 3* will— under the right circumstances— be considered instances albeit less-than-perfect specimen.

In a sense the job of the category is to provide a mini-theory of its instances. That sounds grander than it really is in Seqsee: categories provide a way of describing their instances in a way that facilitates describing relations between instances. For example, in the case of the category *ascending group* Seqsee can describe the *start* and the *end* of instances. For *5 6 7* it means that Seqsee sees the start as 5 and the end as 7, and this “vocabulary to describe ascending groups” helps it find how this group’s relation with *5 6 7 8* can be described.

3.3 Affordances

When we go to a restaurant we know what to do. This was the central example that Schank uses to introduce scripts (Schank and Abelson 1977). According to this theory, we have scripts (think acting) that help us schedule actions.

Another theory that attempts to explain how we know what to do is the affordances theory of Gibson. A chair affords sitting, for example, and that is how we know what to do with chairs. The two theories are not quite at odds with each other: they are just trying to explain similar phenomena at different granularity.

The micro-decisions Seqsee makes in choosing what codelets should be created next occurs via an affordances-like mechanism. Groups afford extending, for example. Even meta-thoughts afford. The thought *I am in a rut* affords destroying weak groups and trying less likely possibilities like exploring if the sequence is interlaced.

Seqsee does not currently use scripts. It is conceivable that something similar may be called for.

3.4 The Stream

Where all this comes together is in the stream of thought. But first a qualification for the word *thought* as used in Seqsee. Thinking about an object in the workspace is just focusing on that object for a short while: calculating its fringe and adding more codelets as are suggested by its affordances. Seqsee has objects (in the workspace) and thoughts about those objects in the stream. Unlike the highly parallel buzz of activity of the codelets, there is something slightly more linear about the stream. When I describe the stream, people sometimes have the sense of one thought leading to the next leading to the next—the $n + 1^{\text{th}}$ thought being caused somehow by the n^{th} thought. That is not how Seqsee's stream works. Either my descriptions have been sloppy, or maybe such linearity was conjured up by the word *stream*. I must describe it more carefully here.

One way to start describing it is to say where the $n + 1^{\text{th}}$ thought can come from. It can of course come from the immediate prior thoughts: The affordance of an object may include focusing on something else, in a manner similar to a street sign affording focusing on the writing. But more importantly the next thought can come from any codelet whatsoever. Consider the hypothetical case where Seqsee is solving five problems in parallel. There are several codelets toiling away on each problem. Since a thought in the stream corresponds to focusing attention on something for a short period of time, it could happen that the thoughts that pass through the stream are well interlaced: thoughts from each of the five problems braided together. In this sense, then, the stream is just a list of all thoughts happening anywhere in the system, sorted by time, and linearity is somewhat of an illusion. It is not totally illusory, though, in the sense that for several simple problems, Seqsee can just go directly to the solution pretty fast. Seqsee solves *1 2 3 4 . . .* blazingly fast because the affordances turn out to be exactly right for this problem.

If the stream is almost merely a birth-registry of attention-fixations, what, if anything, does it buy us? What it gives us is temporality. Seqsee keeps in memory the fringes of the last few thoughts, and these decay with time. If the fringe of a new thought intersects with the fringe of one of the more recent thoughts from anywhere in the system, Seqsee's similarity-ears get perked. In the hypothetical five problem case we would see very strong recency effects and interference. It is as if the stream is distributed throughout the "brain" of Seqsee, but whenever a thought occurs anywhere its echo reverberates throughout, potentially influencing for a short

time the processing everywhere.

Adding a stream was not a completely unmotivated move on my part. While it was done for pragmatic reasons mentioned in the next paragraph, I was pleasantly surprised to recently discover a somewhat similar mechanism used by Dennett (1991) in what he calls the *Joycean machine*—a more or less serial machine simulated imperfectly on parallel neural hardware.

Using the stream has made Seqsee much smarter as compared to Seqsee 12 months ago when there was no stream. In a sequence like ‘1 1 1 1 17 2 2 2 2 18 . . .’ the way Seqsee discovers the 17-18 relationship is by being lucky and thinking about the 17 and 18 somewhat close together in time. Because of all the hubbub of the parallel codelet activity such lucky episodes are guaranteed to happen. Without the stream, seeing the 17-18 relationship was much more unlikely. With a *bond-scout* like mechanism where objects are randomly chosen it is necessary to bias against choosing objects distant from each other because such random long-distance checking is statistically less likely to produce useful results. So even though the 17-18 relationship stands a chance of being discovered it is not predestined.

4 A Tentative Time-Frame

Date	Tasks to be completed
On Oct 6, 2006	Proposal defense.
By Nov 6, 2006	Program capable of seeing interlaced sequences like <i>i</i> and <i>j</i> from section 2.2.
By Dec 31, 2006	The <i>doubler</i> family seen and some progress on self-watching and variation-making.
By Mar 31, 2007	Nearly all sequences in the list seen, along with more progress on self-watching and variation-making.
April 2007 and beyond	Rigorous testing, trying to play around with system parameters systematically; Perhaps Copycat-like lesioning experiments.
Academic year '07-'08	Job hunt, final defense.

References

- Clark, A. (1998). Magic words: How language augments human computation. In S. Boucher and P. Carruthers (Eds.), *Thought and Language*. Cambridge, England: Cambridge University Press.

- Clark, A. (2001). *Mindware: An Introduction to the Philosophy of Cognitive Science*. Oxford University Press.
- Dennett, D. (1991). *Consciousness Explained* (1st ed.). Little, Brown.
- Dennett, D. (1994). Learning and labeling: Commentary on A. Clark & A. Karmiloff-Smith. *Mind & Language* 8, 540–548.
- Hofstadter, D. (1997). *Le Ton beau de Marot: In praise of the Music of Language*. New York, N.Y.: Basic Books.
- Hofstadter, D. R. (1985). *Metamagical themas : questing for the essence of mind and pattern*. New York: Basic Books.
- Hofstadter, D. R. (1995). *Fluid Concepts & Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books.
- James, W. (1890). *The Principles of Psychology*. Dover Publications.
- Mahabal, A. (2006, Sep). Recreation of simon kotovsky's program. <http://www.cs.indiana.edu/~amahabal/simon.pl>.
- Marshall, J. B. (1999). *Metacat: A Self-Watching Cognitive Architecture For Analogy-Making And High-Level Perception*. Ph. D. thesis, Computer Science Department, Indiana University, Bloomington.
- Meredith, M. J. (1986). *Seek-Whence: A Model of Pattern Perception*. Ph. D. thesis, Computer Science Department, Indiana University, Bloomington.
- Mitchell, M. (1990). *Copycat: A Computer Model Of High-Level Perception And Conceptual Slippage In Analogy-Making*. Ph. D. thesis, Computer Science Department, Indiana University, Bloomington.
- Reddy, D. R., L. D. Erman, R. D. Fennel, and R. B. Neely (1973). The HEARSAY-II speech understanding system: An example of the recognition process. In *Proceedings of the International Joint Conference of Artificial Intelligence*, Stanford, pp. 185–194.
- Schank, R. C. and R. P. Abelson (1977). *Scripts, Plans, Goals and Understanding: an Inquiry into Human Knowledge Structures*. Hillsdale, NJ: L. Erlbaum.
- Simon, H. A. and K. Kotovsky (1963). Human acquisition of concepts for sequential patterns. *Psychological Review* 70(6), 534–546.