SEQSEE:

A COGNITIVE ARCHITECTURE FOR SEQUENCE PERCEPTION

ABHIJIT MAHABAL, INDIANA UNIVERSITY

GOALS OF THE PROJECT

My doctoral work is concerned with a set of ideas about high-level perception, situational pressures, contexts, labeling, concepts, categorization, memory, fringes, and the stream of thought. As a core part of my work, I produced a computer program that is able to extend a wide range of cognitively interesting integer sequences in a humanlike way. The dissertation describes these ideas, the architecture of the computer program, and compares its performance to the performance of human subjects on some sequences.

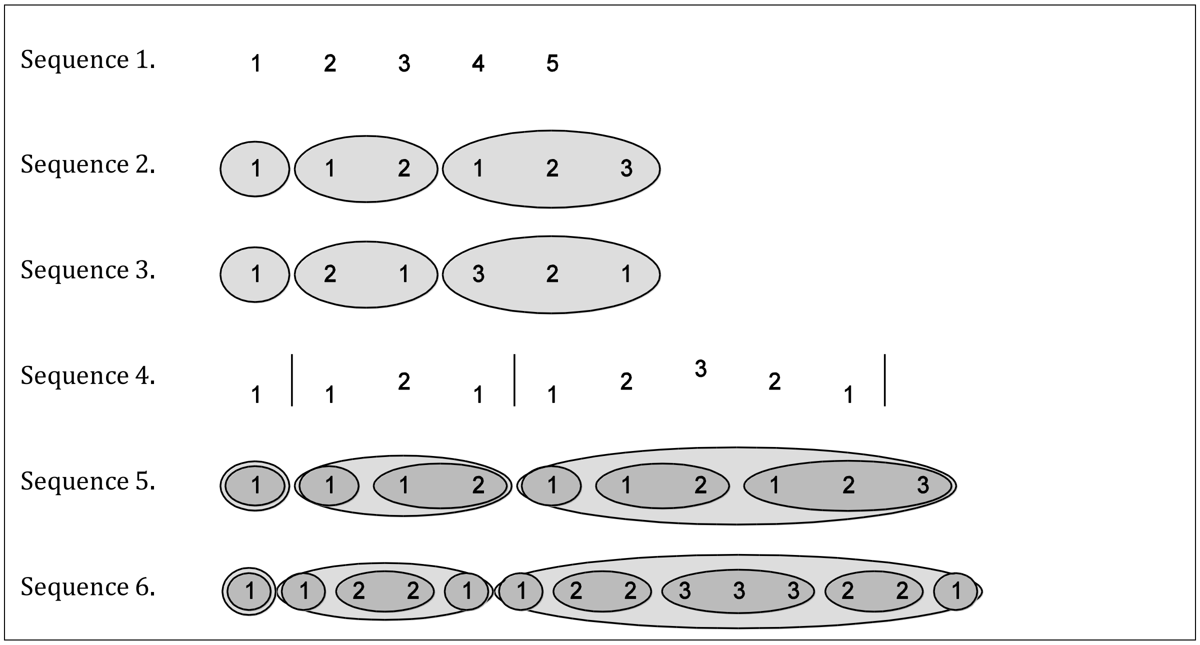
The goals of my doctoral work can be described at two distinct levels. At one level, the goal of the project was to design and implement a computer program that can extend integer sequences intelligently, and the project has in fact resulted in the creation of the program named "Seqsee" (pronounced like the word "sexy"). At a deeper level, however, the goal was to explore human cognition by creating a computer model of activities that require intelligence, always making sure to avoid using shortcuts that are available with computers but are implausible in people. This desire to avoid such shortcuts greatly constrains the solutions that are deemed acceptable. In the absence of these very special and strict constraints and biases, a project whose goal was "a computer program to extend integer sequences" would have developed quite differently.

Seqsee is one more program in the long procession to have emerged from the Douglas Hofstadter's Fluid Analogies Research Group (FARG), over the past quarter-century. These programs[1] tackle a wide range of activities, including solving letter-string analogy problems (Marshall, 1999; Mitchell, 1990), solving Bongard problems (Foundalis, 2006) and designing gridfonts (McGraw, 1995; Rehling, 2001). The common thread linking them all is the creation of computer programs that ideally can perceive and understand complex situations in a human-like way. Seqsee's architecture bears a strong family resemblance to these computer programs.

Without further ado, let me illustrate one of the cognitively challenging aspects of understanding integer sequences. The table below contains six sequences, each starting with a "1". Please note that these are sequences of integers and the ovals and vertical bars are provided merely to speed up the comprehension of these sequences.

[1] This list (sorted by the year the work was completed in) includes Jumbo (Hofstadter, 1983), Seek-Whence (Meredith, 1986), Numbo (Defays, 1988), Copycat (Mitchell, 1990, 1993), Tabletop (French, 1995), Letter Spirit (In two parts: McGraw, 1995; Rehling, 2001), Metacat (Marshall, 1999) and Phaeaco (Foundalis, 2006).

Table 1 Some sequences starting with "1" (Seqsee solves the first five of these)


Sequence 1. 1 2 3 4 5
Sequence 2. 1 1 2 1 2 3
Sequence 3. 1 2 1 3 2 1
Sequence 4. 1 1 2 1 1 2 3 2 1
Sequence 5. 1 1 1 2 1 1 2 1 2 3
Sequence 6. 1 1 2 2 1 1 2 2 3 3 3 2 2 1

The meaning of the initial "1" is radically different in each case, ranging from the simple "the numeral 1" and "a degenerate case of an ascending group" in the first two sequences to the esoteric "a degenerate ascending group of ascending groups" in the Sequence 5, and "a degenerate case of a top-heavy mountain" (see Sequence 6 to know what I mean by that phrase). It is very easy to create sequences where the lowly "1" needs to be interpreted in even zanier ways.

As these sequences demonstrate, any part of a sequence may potentially be very complex. However, people somehow manage to keep things simple unless complexity is called for. For instance, the notion of "a top-heavy mountain" is completely absent when attempting to understand the very trivial first sequence in the table above, and yet, when needed, it can be wheeled out (or perhaps constructed).

Any attempt at implementing a computer system that understands sequences of this complexity in a human-like way must immediately grapple with the following conundrum: how to make it possible for the system to access (or create) complex concepts when needed without gumming up the works with uncalled-for complexity. This is, if I may use this grandiose phrase, the quest for the mechanisms of intuition.

In order to drive home this point about managing complexity, I wish to briefly touch upon a computer program named "Superseeker" (Sloane, 2007), whose goal is seemingly identical to that of Seqsee but whose approach is actually profoundly antithetical to it.

Superseeker is a part of the *Encyclopedia of Integer Sequences* (Sloane, 2007; Sloane and Plouffe, 1995). To use Superseeker, an email must be sent to *superseeker@research.att.com*, containing a line such as "lookup 1 5 19 69". Superseeker replies to the email with possible interpretations of the terms. Just to be clear, I must state here that the following discussion is not intended as a criticism of Superseeker. Superseeker's goal, although superficially similar, is actually profoundly different from that of Seqsee, and its techniques may well be appropriate for that goal.

In order to make sense of the input —"1 5 19 69" , in this case — Superseeker applies 115 distinct transforms to it in order to obtain new sequences that it then looks up in its gigantic 100000-sequences encyclopedia[2] (Sloane, 2007). A few of these transforms are shown below. The third column shows how the transformed sequence $h(n)$ is obtained from the original sequence $g(n)$ and the fourth column shows what applying this transform to the sequence under consideration produces.

Table 2 Some transforms used by Superseeker

Number	Name	Definition	Transformed g(n)
T040	Add 1	$h(n) = g(n) + 1$	2, 6, 20, 70
T041	Subtract 1	$h(n) = g(n) - 1$	0, 4, 18, 68
T018	Take differences	$h(n) = g(n+1) - g(n)$	4, 14, 50
T111	Möbius inverse	$h(n) = \sum_{d\mid n} g(d)\mu\left(\frac{n}{d}\right)$	1, 4, 18, 64
T082	Subtract factorial	$h(n) = g(n) - n!$	0, 3, 13, 45
T010	Divide by factorial	$h(n) = \frac{g(n)}{n!}$	1, 2.5, 3.166, 2.45

Superseeker's strategy is nothing like what a person would do. We do not blindly go through a list of things to try, one by one. In computer science lingo, a strategy like Superseeker's is called *brute-force*. The transform "divide by factorial" underscores the brute force used by Superseeker. Let us ask ourselves if a person would try this transform on this particular sequence (i.e., on "1, 5, 19, 69"). The sequence that would be generated on applying the transform does not even consist of integers, and would by definition be missing from the *Encyclopedia of Integer Sequences*. Thus, in hindsight, the transform had no chance whatsoever of success. A mathematician won't blindly try to apply this transform. What might prompt an attempt to use this transform is observing, for instance, that the ratio between subsequent terms increases as we move right. People have the foresight to avoid the waste of time of applying this particular transform to this particular sequence, and it is worth asking what the nature of that foresight is and how it might be captured in a computer program. This is a central question confronted by the work presented here, and we will encounter it repeatedly.

SEQSEE'S ARCHITECTURE

In this short précis, I only have space to mention salient features of the architecture. Two aspects of Seqsee's working are worth noting at the outset: its blackboard architecture and its nondeterminism. The blackboard architecture is a way for hundreds of entities to collaborate. I would draw an analogy to the way that a police department working on a murder might coordinate its work. Officers must work somewhat independently of each other and yet must also coordinate closely. One officer may be busy verifying the alibi of one suspect, another may be checking out a different suspect in another part of the city. Through a blackboard is how the officers may communicate: each will write down significant findings on the board and read what others have written. What they read will probably influence the leads that the various officers follow next. Thus,

[2]It also attempts to fit a polynomial to the data and to apply other heavy-duty tools, such as trying to represent various types of generating functions as hypergeometric series. See Sloane (2007a) for the full list.

the team as a whole can stay on track and in synchrony, even without any two officers ever meeting face to face.

In the case of Seqsee, determining what the sequence is is the mystery to solve. Significant findings include some piece of the sequence that has started to make sense or perhaps some discovery about how two pieces of the sequence are related. The component called the Workspace corresponds to the blackboard, and this is where observations about the sequence are noted.

The collaborating entities in Seqsee's case are called codelets. A codelet is a tiny piece of code with a small, local impact on Seqsee's internal world. A single codelet, for example, may create a link between two elements in the sequence if it notices that they are analogous. Another codelet may attempt to extend the ascending group "2 3 4", which some other codelet has constructed, by looking for a "5" to its right or a "1" to its left. What a codelet does depends on what aspects of the sequence have already been noticed and noted by codelets that have already run.

The effect of a single codelet is so small that solving all but the simplest sequences requires at least several thousand codelets' joint effort. Solving even moderately complex sequences can require the services of tens of thousands of codelets. Moreover, such tasks as explaining the solution once it is found are carried out not by one but by about a dozen different types of codelets.

At any given point in time, a number of codelets are waiting to run in a staging area called the Coderack. With each codelet is associated a number called its urgency. One of the waiting codelets is chosen to run next, with higher urgency codelets being likelier to be chosen next to run. A consequence of randomly picking the next action is that even on the same input, Seqsee's understanding may follow extremely different trajectories and this can sometimes lead to very distinct interpretations of the sequence.

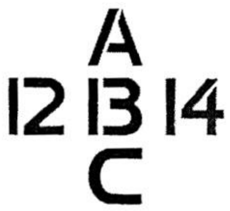
It should be obvious that the immediate next step the program takes, although not completely predictable, is restricted by what codelets are waiting to run. It is fair to say that a chief concern of my doctoral work was a set of techniques for controlling the types of codelets on the Coderack, this being the key determiner of what the program tries in its attempt to understand the sequence. This "demographics of the Coderack" is closely allied to the notion of "mental pressure", to which we turn next.

MENTAL PRESSURE — WHAT SHOULD WE DO NEXT?

The notion of mental pressure may be described by considering the thought processes of a scrabble player who is deciding what word to construct next. Issues that have a bearing on the next move include what letters are already in place and in what configuration and where they are placed relative to open spots containing "triple word scores" and other such a high scoring opportunities. A high score at the next move (by constructing the word, say, "renaissance") must be balanced against, among other things, the opportunities that it opens up for the opponent to score big and against the loss of our own subsequent opportunities from having used up the versatile letters. Any potential move is likely to have a combination of both reasons for and reasons against making that the next choice. It is as if the situation is exerting multiple, incompatible pressures.

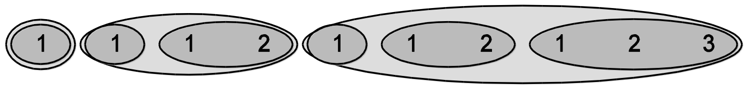
I must point out an obvious but important fact, namely, that it is not the situation itself that is exerting the pressure — instead, it is the perception of this situation. Two different players facing the exact same situation will experience a different constellation of pressures depending on what they notice (and on what they *can* notice, given their experience and knowledge). As an example, if one player is completely unaware of the fact that it is beneficial to hang on to the letters 'A', 'E', 'I', 'N', 'R' and 'S' as these significantly raise the likelihood of scoring a "bingo", this fact will exert no pressure whatsoever on their next move.

The notion of mental pressure is equally relevant to thinking — the mostly unconscious decision of where to focus attention next is just as much under the mercy of dozens of contradictory pressures. To pick a random example, imagine the process of solving a problem in plane geometry. Observations that have a bearing on what aspect of the problem to focus on or what area of the figure to stare at include such things as "these two angles look equal" and "this problem looks very similar to that problem I solved yesterday". In fact, even the act of perceiving a single letter in a written word may be influenced by contradictory pressures, as happens in the famous case of the following figure, where there is pressure to see the entity in the center as "B" as well as pressure to see it as "13".



In designing a computer program to understand integer sequences, I had to grapple with two issues related to pressure. First, how the program would represent and manage multiple pressures. In this regard, I don't believe that I have done anything significantly beyond what was already present in such programs as Melanie Mitchell's Copycat — in both cases, pressure corresponds to the codelets waiting to run. Second, how the program should generate pressures in a way that would guide it towards the solution while avoiding misleading pressure. For instance, in Sequence 5 (repeated below), there should be pressure to see the initial "1" as an ascending group of ascending groups, but pressure to see any other "1" in this unusual fashion would be counter-productive.

Sequence 5.



Where Seqsee has made the most progress, in my opinion, over and beyond what Copycat had already achieved, is in some novel mechanisms for generating very specific pressure.

The following list provides a flavor of the types of pressures within Seqsee.

1. Pressures induced by recognizing structures in the sequence — such as the groups "(1) (2 2) (3 3 3)" and "(2) (3 3) (4 4 4)" — as analogous,
2. Top-down pressures induced by a specific hypothesis about the sequence based on a compelling interpretation of some part of the sequence,
3. Top-down pressures induced by generic hypotheses about the type of structures in the sequence (for example, if many ascending groups such as "1 2 3" have been seen, pressure is generated to look for more such groups),

4. Pressures based on being reminded of similar structures previously seen (whether in sequences seen previously or earlier in the same sequence — for example, if the sequence "(1 1) (2 2) (3 3)" has been seen previously, and a "(5 5 5)" has been seen in the current sequence, there is pressure within Seqsee to explicitly look for a "(6 6 6)" to the right of the "(5 5 5)"), and
5. "reflex" pressures based on affordances (for example, whenever a group has been identified within the sequence, it is a good idea to try and extend it in either direction. Put differently, groups afford being extended, or, equivalently, whenever Seqsee sees a group, its reflex is to try and extend it).

We now turn to one of the novel mechanisms in Seqsee for generation of precise pressures.

THE STREAM OF THOUGHT AS A MECHANISM FOR PRESSURE GENERATION

Consider the following pair of sentences and think about how each word strikes us as we read it:

- Experience consists of experiencing what we wish we had not experienced.
- If Cleopatra's nose had been shorter, the face of the whole world would have been changed.

When we read the first occurrence of the word "experience" — that is, before the rest of the sentence has been read — there is nothing unusual to notice. Many perfectly ordinary sentences start out with this word. What happens, though, is that every word slightly alters the context that subsequent words are read in, so that when we read "experiencing" we cannot help but feel that we have just encountered this word, and perhaps we may ask ourselves if its meaning differs from that of "experience". By the time we reach "experienced", the deck is stacked against our understanding this as an ordinary, vanilla, run-of-the-mill instance of that word.

The three words — "experience", "experiencing", and "experienced" — are almost identical, and this explains how reading one slightly alters the perception of others, but much less than blatant repetition is necessary for this effect. In the second sentence, the word "face" reminds us — I hope the reader will allow me to use the verb *remind* to refer to events less than a second ago — of the word "nose" that was just read, and some effort is channeled to understanding how the two are related in this sentence.

Programming a computer to be sensitive in a human-like way to what was perceived recently is challenging. When the word "nose" is seen, for example, the computer would need to modify the context somehow, but what aspect of "nose" might ring similar to subsequent words is not clear: any of the following words might appear related to "nose": "eyes", "nasal", "smell", or "cosmetic surgery". Likewise, when Seqsee has just seen the group "1 2 3", any of the following groups should remind it of this recently seen group: "1 2 3 4" (because it, too, is ascending), "(1 1) (2 2) (3 3)" (because it, too, is an ascending group in its own way), and also "1 7 3" (because it, too, is of the form "1 x 3"). No, wait, I take that back: "1 7 3" should only appears similar if several groups of the form "1 x 3" have already been seen and the concept has been identified as potentially relevant. Thus, the very notion of what similarities are perceived needs to shift over the course of a single run.

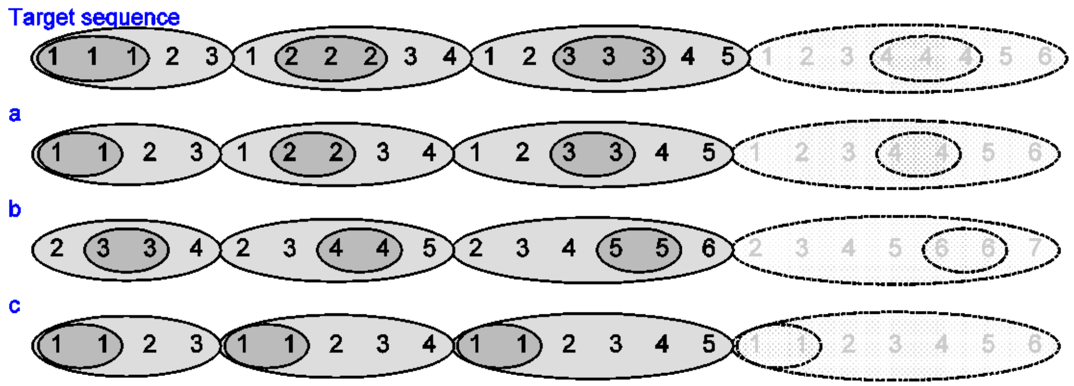
Seqsee uses William James' idea of the stream of thought and his notion of a fringe of a concept to implement this temporal perceptual similarity. For object that Seqsee focuses on, it stores a weighted set of related concepts as the fringe. The fringes of the 10 most recent acts of focusing are stored. Whenever an entity is focused upon, the overlap of its fringe with the stored fringes suggest what recently seen objects may be similar.

The fringe of the object also contains the categories that Seqsee has discovered the object as belonging to. The process by which categories are added to objects is not automatic, instead being one of discovery. Just as Seqsee does not automatically try dozens of transforms on the sequence, it does not try dozens of potential categories for a given object. Here, again, we come across the same issue: how to intelligently guess which categories may make sense without blindly trying hundreds of rare and unlikely categories. The dissertation describes the techniques that Seqsee uses to address this issue.

Since discovered categories are part of the fringe of an object, the perceived similarity between two objects may change as categories for either are discovered. Two objects appear more similar to Seqsee if they are seen as belonging to the same category. Seqsee has the ability to manufacture categories on the fly depending on the structures seen within the sequence, and objects belonging to categories that are heavily represented within the sequence appear all the more similar to each other, allowing Seqsee to zoom in on the solution.

CATEGORY-BASED LONG-TERM MEMORY

Seqsee has a long-term memory that can remember certain aspects of previously seen sequences and previously seen structures within the sequence currently under consideration. Discovered categories are central to this long-term memory, allowing it to perceive the target sequence in the figure below more quickly if it has previously seen the similar sequences "a" or "b" below, but having seen sequence "c" previously gives it no corresponding speed boost.



Another cognitively interesting ability that Seqsee has is being able to see the sequence in the figure above as "a funny version of (1 2 3)(1 2 3 4)(1 2 3 4 5)…". The ability to see an entity as something else is indispensable to cognition (as has also been argued by the sociologist Ervin Goffman (1986), who makes it the centerpiece of his work *Frame Analysis: An Essay on the Organization of Experience*).

COMPARISON WITH HUMAN PERFORMANCE

In the fall of 2007, in Robert Goldstone's laboratory, I presented 48 psychology undergraduate students with the first few terms of a number of integer sequences, and their task was to provide at least the five next terms.

Every student was shown a total of 40 sequences. The instructions specified that a button was to be pressed after they had figured out what the next terms were, and this opened up an extra box in which to enter the terms.

Here I will only mention a pair of sequences. One of the sequences was "1 7 2 7 3 7 4 7". This was seen by 25 students, all but one of whom answered correctly, taking on average 5.13 seconds to understand it. A different set of 23 students saw the nearly identical sequence "6 7 7 7 8 7 9 7", and the 13 students who answered correctly took on average a far higher 16.87 seconds. The misleading "garden-path" "7 7 7" slows people down — and it slows Seqsee down. It takes Seqsee 4 times as long on the second sequence as on the first.

BIBLIOGRAPHY

Defays, Daniel (1988). *L'esprit en friche: Les foisonnements de l'intelligence artificielle*. Liege, Belgium: Pierre Mardaga.

Foundalis, Harry E. (2006). *Phaeaco: A Cognitive Architecture Inspired by Bongard's Problems.* Doctoral dissertation, Computer Science Department, Indiana University, Bloomington, Indiana.

French, Robert M. (1995). *The Subtlety of Sameness: A Theory and Computer Model of Analogy-Making*. Cambridge, Mass.: MIT Press.

Goffman, Erving (1986). *Frame Analysis: An Essay on the Organization of Experience*. Boston, Mass.: Northeastern University Press.

Hofstadter, Douglas R. (1983). The Architecture of Jumbo. In Ryszard Michalski, Jaime Carbonell, and Thomas Mitchell (Eds.), *Proceedings of the International Machine Learning Workshop* (pp. 161-170). Urbana, Illinois: University of Illinois.

Marshall, James B. (1999). *Metacat: A Self-watching Cognitive Architecture for Analogy-making and High-level Perception.* Doctoral dissertation, Computer Science Department, Indiana University, Bloomington, Indiana.

McGraw, Gary (1995). *Letter Spirit (Part One): Emergent High-level Perception of Letters Using Fluid Concepts.* Doctoral dissertation, Computer Science Department, Indiana University, Bloomington, Indiana.

Meredith, Marsha J. (1986). *Seek-Whence: A Model of Pattern Perception.* Doctoral dissertation, Computer Science Department, Indiana University, Bloomington, Indiana.

Mitchell, Melanie (1990). *Copycat: A Computer Model of High-level Perception and Conceptual Slippage in Analogy-making.* Doctoral dissertation, Computer Science Department, University of Michigan, Ann Arbor, Michigan.

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

Rehling, John A. (2001). *Letter Spirit (Part Two): Modeling Creativity in a Visual Domain.* Doctoral dissertation, Computer Science Department, Indiana University, Bloomington, Indiana.

Sloane, N. J. A. (2007). The On-Line Encyclopedia of Integer Sequences, from http://www.research.att.com/~njas/sequences/

Sloane, N. J. A. and Simon Plouffe (1995). *Encyclopedia of Integer Sequences*. San Diego: Academic Press.