Seek-Whence: A Model of Pattern Perception

A Little History (1977-1980)

After 3 years in graduate school (mathematics) at IU Bloomington and 5 in college teaching, I decided to return to IU for a terminal degree in computer science. I asked my advisor, Dan Friedman, to set me up with something 'more practical' than mathematics. He did, Operating Systems and Info Systems to be exact, but after we chatted for a while he told me about a new and interesting guy joining the faculty, doing work in artificial intelligence and human thought. I knew I was interested in that stuff, enjoyed reading about the brain, perception, language, psychology, and so on. So I signed up for his AI class. On the first day of classes, one of my officemates (Bil Lewis) burst in bubbling and babbling about the great course he was taking, taught by a new guy who was writing a book. Who? "A guy named Hofstadter." The same guy who was teaching my AI class! Operating Systems, with 40 people sharing (fighting over) a single machine, suddenly seemed much less exciting. So I did it, went for the shiny thing, and have never regretted the decision.

Between the two classes, I was treated to exciting perspectives on thought, strange loops, analogy, slippage, . . .well, **GEB** in notebook (*i.e.*, modifiable) form alongside approaches to modeling creative processes in code. Mind-blowing, head-spinning, discussion-filled, creative and challenging, taught in an inclusive and conversational style. Doug would also come down to our grad student offices in the basement of Lindley Hall—the old Kinsey interview rooms—and keep the conversation going. Oh, yes! Moreover, as Dan Friedman's TA, I was emerging from a rigid FORTRAN cocoon to the glorious recursion and freewheeling style of Scheme, frantically trying to learn quickly enough to attack the AI course projects.

The projects were assailably impossible.

In semester 1: a 2-person , timed, competitive Chinese checkers player; a Micro-Planner story generator; and a sequence extrapolator. I think we learned a few things from the latter, which was actually the first one assigned. Our instructions included advice to build in some declarative knowledge, such as primes, Fibonacci numbers, continued fractions, Taylor series, and the like. My program also made extensive use of finite differences and other numerical analysis techniques I had learned in my undergraduate math days. This was a far cry from the eventual goals of **Seek-Whence**. We learned that seemingly simple patterns can be much more difficult to solve than the mechanical output of complex calculations. Or: easy things are hard.

Semester 2 held other challenges. We investigated the problems of story generation and story understanding, sense and meaning, and the importance of rich and interconnected nouns in the face of systems that relegated them to 'picture words' and concentrated solely on verbs. My choice of project here was **Kogent**, a Zen koan generator. I soon learned that in order to make nonsense, one must first know what makes sense. Moreover, to support a rich network of nouns it helps to have what we now call object-oriented programs (then emerging in Simula, with Smalltalk in the offing). Still, it was fun: one might even say ‘enlightening’.

Humans are verbal beings. We encounter something and attempt to attach a word or phrase to it, either as an exemplar of a known category or as a relationship among categories. For example, upon having her first ever bite of doughnut, my tiny daughter asked for more ‘cookie apple’. This certainly points us to the central role of analogy in thought but also reveals that we fluidly formulate concepts pre-verbally and then recall or develop a suitable verbalization. For me, the essential model for demonstrating this process is **Bongard Problems**, which Doug introduced in **GEB** and his AI classes. I would have explored them as a research project, but computers in 1980 did not have the graphics capability to support that work. Fortunately, Harry Foundalis later took up this torch in his fine work on **Phaeaco**.

So, fluid pattern recognition with no verbal baggage or need for graphics; malleable active concepts, slippage and reformulation, sparse representation and display requirements, the failings of our over-mechanized sequence programs; and as hard as any simple thing could be: I decided to work on patterned sequence extrapolation, of course with Doug’s counsel and cooperation. We spent many hours together, and later with Gray Clossman and others trying to pin this one down and explore it as fully as possible. **Seek-Whence** was born.

⊖ Marsha Meredith, 9/28/17

<I have included some hand-written notes from Doug that will give a sense of his thought processes as we tried to develop a system to support inductive pattern perception and reformulation. >

Project: Seek-Whence

Marsha Meredith & Douglas Hofstadter

Seek-Whence is a model of human inductive pattern perception. Its goals are to discover patterns, describe them as effective conceptual structures, and reformulate them in plausible ways when appropriate. Processing is performed by small, independent *task series*, which are stored on a *taskrack* and run in simulated parallel. There is no centralized program controller; order must emerge from task effects and interactions. Each task operates in one of three arenas: conceptual (platoplasm), short-term analytical (socratoplasm), or perceptual (cytoplasm). Seek-Whence will settle once it has created and displayed a working hypothesis for the pattern.

The domain of interest is non-mathematically-sophisticated patterns expressed as sequences of nonnegative integers, presented one term at a time. A reasonable pattern is one that relies on nothing more mathematical than successorship and predecessorship. For example:
1 2 2 3 3 3 4 4 4 4 . . .
2 1 2 2 2 2 2 3 2 2 4 2 2 5 2 . . . .
6 3 7 3 8 3 9 3 . . .
'Unreasonable' patterns would be:
2 3 5 7 11 . . . , because 'primes' is too mathematical
5 12 89 147 . . ., because 'get bigger' is too amorphous with no canonical next term.

In practice, a **presenter** creates a reasonable pattern construct and then uses it to enter terms one-by-one as requested by Seek-Whence, the **solver**. As each term is entered the solver attempts to reconcile that new term with its emerging structural model of the pattern. If the model holds, the solver requests another new term. Should the model fail, the solver must reformulate it until it is predictive. Once Seek-Whence has sufficient evidence in support of its model, it will present a summary of that model's structure as a **hypothesis** and supply the next block of terms predicted by it. If correct in the presenter's eyes, Seek-Whence wins. Otherwise, it must return to the reformulation stage and try again.

Reformulation is not simply starting over from scratch. The powerful presence of a hypothesis provides a starting place for evidence-based restructuring. Bonds between terms may exert pulling or pushing pressure that had been overridden in choosing the current hypothesis but now might provide a starting place for necessary restructuring. For example, the initial sequence 1 2 1 2 3 may well have suggested **Successor-group** organization: (1 2) (1 2 3), with predicted extension (1 2 3 4). However new data, extending the initial sequence to 1 2 1 2 3 2 invalidates that hypothesis. The bond between the two 1's might now achieve a higher weight, perhaps leading to a **Symmetry-group** organization: (1 2 1) (2 3 2).

That concept would predict (3 4 3) as the next block of terms, as well as explaining the terms already seen. Seek-Whence now has a newly-reformulated hypothesis.

The concept space for Seek-Whence is based upon eight primitive concepts : Constant, Countup, Copy-group, Successor-group, Predecessor-group, Symmetry-group, Tuple, and Cycle. These can be combined to provide a wide range of sequence organizing concepts.

The Code

Seek-Whence was written in Franz Lisp (Opus 38.69, June 1983), developed at UC Berkeley.
The Franz Manual: “a document in four movements” was written by: John K. Foderaro, Keith Sklower, and Kevin Layer.

The code itself is presented below in five sections (as I am not a musician), organized by level of abstraction or proximity to the input data. The highest level of abstraction is called the **platoplasm**, which (of course) contains our ideal types, such as **Copy-group.** Below that is the **socratoplasm**, a working area of proposals and questioning. Its central structures are called **gnoths** (pronounced ‘knots’ or ‘nots’, and derived from “gnothi seauton” , or “know thyself”, the motto of the Socratic school of philosophy). These gnoths arise from bottom-up noticing and processing performed in the **cytoplasm**, our bottom and most grounded level, populated by **glints** and **gloms**. Each glint is a representation of a sequence term that has been entered by the user. Glints are atomic and undissolvable gloms. Larger gloms are created as noticer processes discover interesting relationships between gloms and **bond** them together. The gnoths notice interesting gloms and formally tie things together in order to play with them. Get it? Gnoths tie things together. . .?
It is important to note that, while processing begins in the cytoplasm, Seek-Whence is not a purely bottom-up system. All noticings, recombinations, and results associated with the higher levels percolate down to drive or terminate further work in the cytoplasm.

bonds.l
This section contains the all-important (startup) function that sets up processing , looks for relevant input and sets out appropriate tasks to inspect any new data and relate it to the old. Noticings of import cause **bond**s to be set between elements and element groupings.

gloms.l
Sufficiently strong or numerous bonds awaken **glom-scouts** that inspect connections to see whether or not the items in question should be acknowledged as

a **glom**, a somewhat sticky ball of rice. A flurry of activity surrounds such interesting groupings as some processes make connections and other attempt to break up groupings. Importantly, the system as a whole is moving into the realm of primitive concepts. **Plato-scout**s investigate to see if any gloms have a familiar structure. In human terms, the system has noticed something and will try to get a handle on it. Gloms are ephemeral by nature, but strong ones that survive may serve as a proposal for organizing the data. If a **Plato-scout** confirms that a glom does appear to be related to an ideal type, a **template** of its structure may be created to prolong its lifetime and give it a chance to be perused by a **Plato-evaluator.** It is now time for a more global look at the template to see if it is effective at describing the sequence pattern thus far.

gnoths.l
A template can temporarily protect its underlying gloms from destruction until an **evaluator** either patches it, rejects it, or pushes it up to a **gnoth.** If the latter occurs, a storm of activity ensues. A gnoth that is deemed successful at representing the sequence's pattern in converted into a **hypothesis**, ready to be tried out on new evidence. A new term is called for, and if it fits the hypothesized pattern, Seek-Whence will venture its guess at the underlying structure. All is good! . . .unless it is WRONG! An incorrect guess will lead all levels to a process of **reformulation**.

reforms.l
Back to the drawing board? Not exactly.

A hypothesis has two components: a **form** or encapsulation of a pattern, and a **box** or active realization of the pattern. When a box is **hit** it produces the next value predicted by the hypothesis. The form can be presented to a user as the system's guess at the overall pattern; the box is used for prediction or testing .

When a hypothesis fails to predict correctly or is rejected by the user as the correct pattern encapsulation all levels of processing are engaged in reconsidering and reformulating it. Any tensions within or among gloms that might have been suppressed in deference to the **reigning class** (the Platonic class at the front of the hypothesis form) are now able to assert themselves more forcefully. The possible consequences include both intra-glom and inter-glom restructuring as the system **slips** from one formulation to another. When reformulation activity abates, the current favorite hypothesis is readied for evaluation and testing and the system prepares for more evidence or announces its new guess.

Sample Run

<dissertation pp. 170-179 gives a verbose view>

References

Hofstadter, D.R., Clossman, G.A. and Meredith, M.J. (1980). Shakespeare's plays weren't written by him, but by someone else of the same name. Technical Report No.96, Department of Computer Science, Indiana University, Bloomington.

Hofstadter, D.R., Clossman, G.A. and Meredith, M.J. (1982). SW: A computer model of perception, abstraction, and induction. Internal memo, Department of Computer Science, Indiana University, Bloomington.

Meredith, M.J. (1986). Seek-Whence: a model of pattern perception. Technical Report No. 214, Department of Computer Science, Indiana University, Bloomington.

Meredith, M.J. (1991). Data modeling: a process for pattern induction. *Journal of Experimental and Theoretical Artificial Intelligence,* 3(1), pp. 43-67.

Meredith, M.J., Reynolds, P. and Wehking, A. (1983). Pattern perception experiment. Presentation to the Illinois State Academy of Sciences, Computer Science Section.