HOW TO SUCCEED IN BUSINESS

William Bricken
June 1989

I do have an opinion about how to succeed in research with LoF. Unfortunately, it is not a particularly pleasant path. We practice a mathematics that is both esoteric and non-traditional. Almost all academics are paid to maintain the quality of existing knowledge. When a new idea comes along, it is the responsibility of the new idea to justify its worth, without costing great effort on the part of the traditionalists.

For example, I've received lots of mail from crank mathematicians. I ask myself "How much time should I spend trying to understand this stuff?" The only sensible answer is "Virtually none." So we can't expect an academic to study LoF for two years in order to see what we have found in it.

What we must do is prepare detailed and immaculately correct maps from their language to our language. I returned to graduate school and studied logic and computer science. While there I talked LoF theory and consequently learned the premise: we must prove the worth of LoF by improving on existing techniques.

This is not a particularly philosophical perspective. I'm not working in the domain of words which change other people's minds, I'm working with products that cause people to give me money. So I need to be better than the competition at useful tasks.

After thoroughly learning the perspective and the vocabulary of traditional approaches, I made maps. In particular, I wrote computer programs that transcribed traditional problems (such as circuit verification, theorem proving, and program visualization) into LoF, computed using LoF, and then returned the answer in traditional notation. From the perspective of a user, LoF was completely invisible. It's just that when the LoF algorithms were being used, the answers returned an order of magnitude faster.

My contribution was not to do something that no one else could do, rather it was to demonstrate a more efficient way of doing the same old thing.

There are many traditional domains that would improve with a LoF perspective. The important point is that people (at least business people) don't care about theory and proof and axiomatization. They care about demonstrable results. If you are interested in fermion physics, I would suggest you take a traditional degree in that subject, and use your courses to make maps between how they think and how we think.

One benefit of this path is that you learn very quickly that LoF improves almost any foundational approach, but becomes irrelevant very quickly as the foundation is elaborated. Varela stopped using LoF for this reason. To make

LoF useful (practically not philosophically) is quite difficult, requiring knowledge of the subtlies of two systems and construction of a map across both foundational and elaborated aspects.

I do not know of others who have found commercial use for LoF, and I have no entirely succeeded in that myself. Although my work has justified a position for myself, LoF is not currently in any money making product. That I can survive in business is not so much due to my LoF work, but to my formal training in the other disciplines. The competition for research positions in business is tremendous. A PhD is expected.

Please understand that I share with you these rather blunt facts in order to describe accurately how to succeed in business. At the personal level, it took me nine years of very hard work to move from amateur to professional LoF mathematician. As another example, recall that Kauffman is known first for his work on traditional knot theory.

There is another essential step in establishing credibility: one must publicly distribute work of high quality. Professionals come to a decision about one's work by studying it. The kiss of death is to say that one has excellent work that cannot be shared. Spencer-Brown has done himself a great disservice by not continuing to publish. The idea needs sustenance in order to grow. And sustenance comes from mass exposure. A friend of mine invented the concept of hypertext thirty years ago. He adamantly extolled the idea for 27 years before it caught on. He never abandoned it, never backed off, never kept quiet.

So you must make you work available to everyone who is interested. This, of course, requires that it be assembled into a paper ready for distribution.

We have available an intermediate course, one which I am most interested to pursue. Spencer-Brown and yourself hold a treasury of information. I am quite good at identifying application areas, particularly in computer science. Computer science is a mathematician's paradise, it aches for new and better algorithms. Computer science is also a mathematician's hell, it is constructivist and finite. Abstract existence proofs are not at all useful, semantic linkages are mandatory.

What is missing is the communication of ideas. And here we have a good start. In particular, you have about half of my written work. The deductive engine is valuable. But the key point is that it is the computer code that is valuable, not the mathematical idea. In the US (and Europe), a mathematical idea cannot be patented or copyrighted. Its implementation can.

I discovered this rather rude fact directly trying to profit from my mathematical knowledge of LoF. The summary is that academics can claim ideas, they get rewarded with status and reputation. Businesses can claim

implementation of ideas, they get rewarded with money. In general, you can take your pick, one or the other.

So I am keen to contribute to the network of interested mathematicians. But, to my knowledge, that's us. I long for comment on my work, I suffer without a mathematical editor. It's really quite strange. The Losp paper is well known, read by hundreds, yet no one can comment on the mathematics. I have talked to many folks, and the consensus is that it works (syntactically) but it is incomprehensible (semantically). It takes years of concentrated mental effort (as you must know) to move one's thoughts from a token-oriented symbolic model to a distinction-oriented boundary model. It implies literally seeing the world differently, and this difference, once internalized, significantly impacts one's ability to interact with others.

Isn't Lou's relativistic distinction work wonderful! I too take joy in sharing these concepts. I agree with you that we are forming the mathematical foundation of the next century. I should hope our interests might grow into a working group. I believe it is a little too early to establish a formal group, but I'd certainly contribute to such an organization when it reached, perhaps, a half-a-dozen active contributors.

Your description of the deductive algorithm is accurate. There is a slight generalization in rewrite engines: the sequence of application of valid transformation rules is controllable. Generally, there is a Match-Resolve-Alter cycle. Every initial is first matched to the working expression, resulting in a list of possible reductions. Then one reduction is selected, any selection criteria can be specified. That reduction is applied, resulting in a new expression and a new reduction cycle.

This architecture is quite general. You could provide a strict reduction sequence (say, clarify or extract, and if extract then absorb) as the selection criterion, or you could give priority to a particular initial, or you could pick reductions randomly, or you could pick them asynchronously, as we have done in the parallel engine.

The next step is to recognize that all problems can be stated in logic rather than in control structures. "Clarify or extract, and if extract then absorb" becomes

((C or E) and (if E then A))

This is then LoFfed:

((C E) ((E) A))

This meta-language template drives the engine.

Using your problem of demonstrating E' from E plus initials as an example (= is logical equivalence):

Control structure:

E + initials ==> E'

Logic:

(E & initials) = E'

LoF:

((E) (i)) = E'

Next we convert

A = B

to the logical form

((if A then B) and (if B then A))

to get

(((((E) (i))) E') ((E') ((E) (i)))) = ( )

which simplifies to

(((E) (i) E') ((E') ((E) (i))))

So you just plug whatever E, i, and E' you want into this meta-template and let it chug. Note that i is not an expression, it is the Match-Resolve-Alter process. The meta-template takes care of organizing processes to reach the goal.

The observation you make about

((b) a) ((c) a) = ((b c) a)

has wisdom on many levels. I have struggled with it for years. The solution lies deep with many controversies in formal logic and in computer science.

Computer languages dislike the return of multiple values. To get a list from 1 to 5, I must use an enclosing data-structure, the list, to hold the five values as a single object. In the case of

((b) a) ((c) a)

an implementation will need a grouping device cause there are two forms sharing the top level space. Naming any new grouping device (such as *list*) is equivalent to degrading the LoF void. So we chose to introduce

((A)) = A

as the first axiom of the system, giving:

((((b) a) ((c) a))) = ((b c) a)

which permits transposition and resolves your objection.

The cost of the resolution is to always build systems with the axiom

((A)) = A

Since this has never been a practical inconvenience, we have been happy to restrict our mathematics to get rid of several theoretical problems. Similarly, we have focused on propositional logic, an expressively weak language, in order to assure results.

So we don't need to formally increase tokens (in effect degrading the convergence of the reduction algorithm). We do that just once, at the beginning of the problem, so that we are always addressing an expression that has a single top-level parens.

An analogous solution we have used is to explicitly introduce an observer distinction, so that all expressions are within a single context. Then

((b) a) ((c) a)

becomes

[((b) a) ((c) a)]

where [...] is the observer.

Transposition now works, but at the cost of letting symbols get beyond the scope of the observer. This situation is identical to that of a meta-language, the meta-transformation being

[A] B ==> [((A) B)]

The cost here is one direction of Reflection:

A => ((A))

Read for logic, meta-rules are just valid inference rules. The rule REFLECTION is named appropriately in LoF, it is the formal process of reflection in logic, of moving between theory and meta-theory, between use and mention.

Which is to say, the body of LoF initials that does not include

((A)) = A

is that body that does not permit linkages between symbol and meaning.

Politics

Yes, the Quine-McClusky algorithm is exponential, in that increasing the number of variables exponentially increases the effort required for minimization. This cost shows up as transformations which increase symbolic structure. Boolean minimization is the most important mathematical question in computer science, since most hard problems have been linked to it. A non-exponential algorithm would be far more important than a four-color proof, not only would it amaze the mathematical audience, it would impact a multi-billion dollar industry.

In regard to J and E2. Lou is quite careful about attributing ideas to their originators. That's where I first heard your name. The main criteria for attributing work to others is to be able to cite their publication. The issue is not who thought of it first (since parallel generation of ideas is common), rather it is who published it first. The recent example is table-top fusion, which was prematurely released not because others were also working on the idea, but because others were about to publish their work. Academic referees simple will not recognize attributions that cannot be documented.

I am so glad we have made contact. I am most eager to hear your results in axiomatizations and to encourage both you and Spencer-Brown to release as much as possible. It is by sharing results with others that the idea can grow. I am not particularly smart about mathematical proofs, but I have been able to put the work of others to good use. I've learned that mathematicians gain value by first delivering then waiting.