REVIEW OF A NEW NOTATION FOR LOGIC

William Bricken
October 1985

I have reviewed the materials you sent to me with interest. My top level conclusion is that your work is better suited for educational psychology than for computer science. Of course, I can only judge what you sent me from the perspective of my own training. I'll organize my replies within specific disciplines.

I. Education: obviously, your work has focused on techniques to convey elementary Boolean Algebra to students. I really can't address the quality of these tools, it seems like an empirical question. We both agree to the desirability of clearly presented foundations at an grade school level. (There is talk of substituting formal logic for the calculus as the freshman math course in college.) How this should be accomplished depends on one's assessment of the clarity of existing concepts in logic. Your work seems to adopt the elementary concepts of traditional logic (such as negation, the 16 binary relations, binary arity). Your suggestion to incorporate visual meaning into notation directly addresses teaching.

II. Mathematics: The traditional perspective is that a change in notation cannot affect the structure of a mathematics. Thus, your research does not comment to mathematics other than to better highlight the existent structure of binary propositional sentences.

III. Computer Science: The domain you address is totally solved for CS, usually at the level of hard-wiring the computational machine. That there are 6 or 16 labels is irrelevant since the machine does not distinguish labels, only configurations of wirings. The important point is that what you address is not a source of difficulty for machine logic. There might be an application at the level of software, in that the syntax of a language must be friendly. However the machine places strong constraints on the expressability of a language. It does not care about philosophy or motivation. With this in mind, I'll delineate some problems in CS that you can apply your notation to. I'll also specify the relative contribution of a clean solution to these problems.

A. The complexity of distribution

IF (IF a THEN b ELSE c) THEN d ELSE e

==

IF a THEN (IF b THEN d ELSE e) ELSE (IF c THEN d ELSE e)

Test your notation on these equivalent expressions. Read the IF THEN ELSE structure as (IF a THEN b) AND (IF (NOT a) THEN c). If the form of LA comes out to be identical for both expressions, then you have something. This is a relatively easy, solved transformation problem, but a good test of potential.

If your notation yields different forms for each expression, then we enter the domain of proof, and

B. Simple proof techniques

How many and how complex are the transformation rules to convert one expression to the other? What about convergence of transformations, that is, what is the control structure that guides the transformation steps? Only one transformation rule is great, since there is no branching in the choice of what to apply. As best that I can judge, your concerns are focused on transformational logic, and not on proof or proof theory.

The difficulty with the problem I pose is that it requires selective, and intelligently guided, application of the rule of distribution. All difficult transformation problems in propositional logic have this characteristic. Unfortunately, your papers do not discuss it.

C. The minimization problem

This is a biggie, solving it elegantly will lead to fame. A simple example:

NOT ((a AND b AND c AND d) OR ((NOT a) AND (NOT b) AND (NOT c) AND (NOT d)))

What is the simplest (ie shortest) form of this expression? The first step is to specify a conversion algorithm that actually generates the simplest form. Then show that it is not combinatorially explosive. Finally generalize it to arbitrary expressions.

I would be very pleasantly surprised if you find that your work addresses these difficulties.

D. The unification problem

The above three problems exhaust the interesting problems in propositional logic. The more difficult stuff is in Predicate Calculus, for example:

"Find a fast algorithm that identifies the structural similarity of two arbitrary terms. A term replaces the single letter notation above. It can be arbitrarily complex, such as (a * (+ b c)), in the arithmetic of numbers."

Well, these are the problems I have been working with. I'm afraid my work overlaps only slightly with yours. We do address fundamental issues of representation, but you focus on the man-symbol system and I focus on the machine-symbol system. The requirements of the two are vastly different. Please understand that my interest is in fast (ie elegant) algorithms, and not in easily comprehendable representations. In my work, for instance, I make no distinction about the number of arguments in an expression. The only operator I use is that of DISTINCTION (see Spencer-Brown's Laws of Form).

I wish you well in the incorporation of your ideas in schools.