Verbs, Nouns, and Computer Algebra, or What's Grammar Got to do with Math?

Richard Fateman

December 18, 2008

Abstract

Computer algebra systems (CAS) are supposed to do mathematics. Unfortunately, much of human mathematics presentation and manipulation depends on humans to understand what is going on more generally. This context is often not made available to the computer system, and so it will sometimes fail to make the right guess at the meaning of constructions, unless they are made unambiguous. Among many issues, one that has been raised and resolved, is the need for attributing properties to operators that distinguish between "the name of the operation" and "the operation itself".

As a simple example, referential transparency requires that if $x = y$, then any true statement $P(x)$ about x must be true about y, namely $P(y)$ is also true.

Consider now the well-known equation $\sin(0) = 0$, and the statement $P(r) :=$ "$r = 0$ is a rule to simplify the sin() function". The statement $P(\sin(0))$ is clearly true. Therefore $P(0)$ should be true as well. It is not, since $0 = 0$ is not a rule to simplify sin().

The point here is that sin(0) is not equal to 0 if sin is treated as the name of an operation. Perhaps it is a character string, or a data structure of some sort: a noun phrase. The corresponding expression including the verb form of sin is one in which sin(0) can be simplified to something equivalent, but in some sense preferable. Here it is reasonable to replace it by 0.

1 Introduction to nouns and verbs

Every [check this?] computer algebra system that attempts to mimick reasoning about functions must grapple with a representation issue that separates operations and their names.

If we cannot do so, our ability to communicate relationships is severely limited.

Maple: Derivative vs derivative

Mathematica: D vs Hold[D...]

Maxima: diff vs 'diff

Because Maxima inherits some operations from its earlier days when such niceties were not observed, there are some anachronisms. In particular, evaluation and simplification are sometimes conflated, and sometimes separated. Maple may have some similar problems but we are not going to examine them here.

Mathematica may not have the same problems because evaluation and simplification are replaced by rule transformations. Thus the problems inherent in nouns and verbs become problems on a slightly different terrain, rule sets. In Mathematica, rules are guarded by Hold or HoldAll or HoldComplete or Defer or Unevaluated. It seems unlikely that this is a satisfactorysolution, given the difficulty of using these.

As a simple example, already mentioned in the abstract, if we try to collect equations for simplification of trigonometric functions and display them, one will probably reflect the fact that $\sin(0) = 0$. It is almost impossible to get Maxima to display this equation. The one way I have found is `block([], sin(0)=0)` which works because the body of programs is not simplified or evaluated until the program is executed.

Maxima provides a noun form for sin, namely 'sin, but sin(0) is "simplified" to 0, and so Maxima ignores the user's attempt to refer to the name of the operation (the noun form) by quoting.

In other cases, this works much better.

2 A example that can work

A trick often taught in calculus is "logarithmic differentiation" based on the following situation. Given a complicated expression, say a product of three large expressions:

$$q = r * s * t$$

and the need to compute the derivative of q, consider taking the logarithm of both sides,

$$\log(q) = \log(r * s * t)$$

then expand the right-hand side into $\log(r) + \log(s) + \log(t)$, and take derivatives.

$$D_x \log(q) = D_x(\log(r)) + D_x(\log(s)) + D_x(\log(t))$$

or

$$(D_x q)/q = (D_x(r))/r + (D_x(s))/s + (D_x(t))/t$$

then multiplying both sides by q, we have a rather more compact version of $D_x q$ than otherwise.

Writing this out in Maxima requires that we be able to discuss $\log(a * b)$ without it automatically expanding to $\log(a) + \log(b)$, and $D_x(\log(q))$ without it expanding to an equivalent $(D_x q)/q$. It also is helpful if one can traverse these unexpanded expressions with some meta-operation that makes the various named operations such as log and D_x operate.

Even stating the initial relationship as $q = r*s*t$ presents a minor issue. We must, in Maxima, write out q as $q(x)$ or else declare via `depends([q,r,s,t],x)` that q, r, s, t are all implicitly operators of no arguments that depend on the global variable x, in such a way that the computer system knows that $D_x q$ is not to be evaluated to 0.

We leave to the reader, the exercise of producing each of the above equations as a display in Maxima, and furthermore showing that the final result of logarithmic differentiation is equal to the conventional method from first principles.

3 Screwups in Maxima

It is not possible to produce the noun forms of many operators because the simplifier may have been constructed to take them away. Thus `'sin(0)` is 0 instead of unchanged. Most annoyingly, `'integrate(u,v)` becomes `u*v` even when one is attempting to instruct Maxima about integrating new forms with particular patterns associated with u.

4 Standing on the Shoulders. Not

This noun/verb issue was raised and a solution implemented circa 1972 or earlier in Macsyma. It is unfortunate that the CAS Maple and Mathematica, designed much later, were not built initially with a solution to this problem and were then subjected to ad hoc retrofits.

5 What does Maxima do?

What *does* Maxima do?

6 A clean solution

Is one possible? Explain.